peter bishop

I CALCOLATORI DELLA QUINTA GENERAZIONE

ricerche, strutture, linguaggi

MANUALI SCIENTIFICI

J. Attikiouzel *Pascal con l'elettronica*
Harold Abelson e Andrea A. diSessa *La geometria della tartaruga*
Martin Cripps *L'hardware dei computer*
M. S. Carberry, H. M. Khalil, J. F. Leathrum, L. S. Levy *Fondamenti di informatica*
R. Danese, C. Risegato e F. Perrotta *Simulazioni di fisica in Basic*
Reginaldo Danese *Il calcolo ricorrente*
Emilio Gagliardo *L'analisi matematica*
D. R. Green e J. Lewis *Le scienze con il calcolatore tascabile*
Sergio Garue *Elettronica digitale*
Peter Grogono *Progammare in Pascal*
Peter Henrici *Matematica con il calcolatore tascabile*
Aubrey Jones *Astronomia con il calcolatore tascabile*
Antonio Leone *Il moto dei corpi celesti*
Thomas W. Norton *Gli esperimenti facili: energia solare*
Carol Anne Ogdin *Il progetto dei microcomputer: hardware*
Carol Anne Ogdin *Il progetto dei microcomputer: software*
A. Oldknow e D. Smith *Imparare la matematica con il microcomputer*
E. S. Page e L. B. Wilson *La combinatoria computazionale*
Nicolò Pintacuda *Algoritmi elementari*
Nicolò Pintacuda *Insegnare la probabilità*
Nicolò Pintacuda *Primo corso di probabilità*
Nicolò Pintacuda *Secondo corso di probabilità*
Giuliano Romano *Introduzione all'astronomia*
Manfred Schroeder *La teoria dei numeri*
Andrea Sgarro *Crittografia*
C. J. Snijders *La sezione aurea*
Paolo Toni *Disfide matematiche a scuola*
Vittorio Zanetti *Gli esperimenti facili: fisica di base*
Rudolf Zaripov *Musica con il calcolatore*

peter bishop

I CALCOLATORI DELLA QUINTA GENERAZIONE

ricerche, strutture, linguaggi

franco muzzio editore

Indice

Prefazione

L'obiettivo dello sviluppo della quinta generazione di computer — computer dotati di un'intelligenza maggiore — è, dopo l'invenzione dei transistor, il progresso più significativo che si sia mai compiuto nel settore della tecnologia dell'informazione. Esso lancia una sfida ad ogni aspetto della tecnologia informatica: prodotti, mercati, utenti ed esperti nel settore e, soprattutto, mette in discussione la posizione degli USA e dell'Europa quali principali innovatori. Sia il nome, sia il concetto, sia la linea di sviluppo della nuova generazione hanno origine in Giappone, e il Giappone sta tenendo testa a ogni scommessa fatta sullo sviluppo. Grosse somme di denaro e tempo ed energie di alcuni tra i maggiori professionisti di tecnologia informatica, sia accademici che industriali, vengono investiti in una serie di progetti, alcuni in collaborazione, altri in competizione, per dar vita, negli anni novanta, a una generazione di computer basata su principi che neppure esistevano agli inizi degli anni ottanta. Se un gruppo di connazionali riuscisse a passare in testa nella corsa allo sviluppo, quel paese potrebbe diventare la forza dominante nell'industria informatica per un decennio o più.

Questo libro è stato scritto con lo scopo di colmare il vuoto esistente tra i piccoli gruppi di professionisti attivamente occupati nello sviluppo della quinta generazione ed il resto della comunità cui gli stessi appartengono, costituito da laureandi in informatica e loro docenti, professionisti in tutti i campi dell'informatica, direttori di progetti con componenti di tecnologia informatica ed insegnanti responsabili dei

corsi di informatica. Esso presenta i computer della quinta generazione in generale, ma allo stesso tempo li esamina in modo sufficientemente approfondito da fornire una buona conoscenza generale sulla materia. Non è necessario che il lettore conosca già gli argomenti che hanno preceduto la nascita della quinta generazione di computer — intelligenza artificiale, architetture parallele, linguaggi di programmazione logica. Si presume, comunque, che abbia almeno una conoscenza elementare dei concetti, procedimenti e termini tecnici dell'informatica moderna.
Gli argomenti trattati in questo libro sono i seguenti:

- Breve panoramica storica sullo sviluppo che ha condotto ai computer moderni, in cui ci si sofferma sui concetti base degli elaboratori, molti dei quali vanno riesaminati alla luce del programma della quinta generazione.
- Presentazione, sotto forma di introduzione, dei concetti base dell'intelligenza artificiale e riepilogo dei progressi che si sono compiuti in questo settore fino al giorno d'oggi.
- Descrizione di ciò che si intende per struttura generale di un sistema di elaborazione della quinta generazione, e presentazione dei progetti cui si sta lavorando in Giappone, negli Stati Uniti e in Europa per lo sviluppo della quinta generazione di computer.
- Descrizione delle tecnologie hardware e software necessarie alla costruzione dei computer della quinta generazione.
- Future applicazioni dei computer della quinta generazione, e possibili conseguenze della loro introduzione.

Questo libro non intende fornire un quadro dettagliato dei progressi compiuti all'interno dei vari progetti della quinta generazione. Esso si occupa soprattutto dei concetti e delle tecniche su cui sono basati l'architettura dei computer, l'ingegneria del software, la struttura delle interfacce utente e l'applicazione dell'intelligenza artificiale, punto cardinale dell'evoluzione degli elaboratori intelligenti. Dove sia possibile, si sofferma sulla grossa differenza tra l'attuale utilizzo degli elaboratori e quello che ci si aspetta dalla nuova generazione. Gli elaboratori della quinta generazione renderanno obsoleti non solo i computer attualmente utilizzati in molti settori, ma anche gran parte della cultura informatica tradizionale, soprattutto la programmazione procedurale convenzionale con i linguaggi ad essa associati, e le strutture hardware centrate su architetture sequenziali a processore unico.

Ringraziamenti

Desidero ringraziare molte persone per avermi fornito il materiale bibliografico, gran parte del quale al tempo non ancora pubblicato, per avermi aiutato nelle ricerche effettuate per la stesura di questo libro: John Darlington per le informazioni fornitemi sull'elaboratore Alice, Robert Bailey per il materiale sul linguaggio di programmazione Hope, Meir Lehman per il materiale sull'ingegneria del software e Richard Ennals per i primi documenti relativi al progetto Alvey. Tutti i membri del gruppo di ricerca sulla quinta generazione, dell'Imperial College di Londra. Ringrazio anche John Campbell dell'Università di Exeter e Brian Meek del Queen Elisabeth College per l'approfondita revisione dell'intero testo e per gli utili consigli che mi hanno dato su molti argomenti complessi. Ringrazio per aver potuto usufruire della biblioteca dell'Imperial College, dove sono state svolte gran parte delle ricerche per la stesura di questo libro.

luglio, 1985

Peter Bishop

1

Introduzione

La quinta generazione rappresenta una svolta fondamentale nello sviluppo degli elaboratori elettronici digitali. Si scosta dall'obiettivo tradizionale, che risale a circa 35 anni fa, per la cui realizzazione ha lavorato l'industria informatica, che consisteva nel conseguire il "più piccolo, più veloce, più economico". Le prime quattro generazioni di computer sono, essenzialmente, variazioni sul tema originale — una macchina in grado di elaborare automaticamente dei dati, controllata da un programma memorizzato. Lo scopo della quinta generazione, invece, è realizzare macchine "più intelligenti".

I computer della quinta generazione sono, comunque, soprattutto il risultato di quella stessa serie di eventi che ci ha condotto al piccolo computer da tavolo e al missile da crociera terra-terra. Questo capitolo presenta brevemente un resoconto di questa sequenza di fatti soffermandosi soprattutto su quegli sviluppi che stanno acquistando un nuovo significato alla luce dei nuovi eventi. All'interno di questo processo identifica alcuni concetti di base che si devono conoscere per comprendere il significato della quinta generazione. Si tratta, di necessità, di una generalizzazione, in cui vi sono molte semplificazioni e si attribuiscono a pochi personaggi più importanti i concetti che, di fatto, sono scaturiti dal lavoro di gruppo — soprattutto da quella stessa collaborazione a livello accademico, politico e militare senza precedenti che ha prevenuto la sconfitta degli Alleati nei giorni peggiori della Seconda Guerra Mondiale, e ha contribuito al rapido raggiungimento della sua conclusione.

1.1 Le origini degli eleboratori elettronici digitali

Il concetto di calcolatore programmabile, come lo intendiamo al giorno d'oggi, fu elaborato per la prima volta da Charles Babbage (1791-1871) mentre stava lavorando al progetto di quella che chiamò Macchina Analitica. Dal 1834 fino alla sua morte, Babbage perfezionò le sue idee e i suoi progetti per la realizzazione di un calcolatore di uso generale controllato da una sequenza di istruzioni (Hyman, 1982). La macchina aveva un'unità operativa, una memoria centrale, dispositivi per l'input e l'output dei dati e per il controllo dei cicli di elaborazione di quello che noi oggi chiameremmo un programma (Babbage, 1837). Questi concetti venivano realizzati con mezzi meccanici, utilizzando complessi rotismi, secondo le tecnologie dell'epoca. Comunque, anche con le migliori tecniche di fabbricazione e di controllo della qualità era impossibile realizzare le strutture progettate da Babbage. Si riuscirono, infatti, a costruire solo pochi frammenti della Macchina Analitica, cosicché le idee di Babbage non vennero realizzate per quasi un secolo. Esse costituirono le basi su cui fu costruito il primo (e forse ultimo) calcolatore elettromeccanico, il calcolatore automatico a controllo di sequenza (ASCC), completato nel 1944 (Ashurst, 1983), e influirono sulla struttura dei calcolatori elettronici.
Anche se non venne mai costruita alcuna Macchina Analitica, si discusse molto su quali fossero i limiti delle sue capacità e fino a che punto il suo comportamento avrebbe potuto essere definito intelligente. Le osservazioni più acute vennero fatte da Ada Byron, secondo cui "La Macchina Analitica non pretende di creare nulla. Essa può fare qualsiasi cosa nella misura in cui noi sappiamo ordinarle in che modo farlo" (Menabrea, 1843).
Il periodo in cui si lavorò con maggior intensità allo sviluppo degli elaboratori elettronici digitali furono gli anni tra il 1936 e il 1946. Nel 1936 Alan Turing pubblicò uno scritto in cui presentava il progetto teorico di un computer che non era semplicemente di uso generale — infatti poteva, in teoria, risolvere qualsiasi problema che venisse formulato per mezzo di ciò che oggi chiameremmo un algoritmo. Nel 1946 John Von Neumann e i suoi colleghi pubblicarono un documento in cui delineavano i principi della struttura di un elaboratore elettronico controllato da un programma memorizzato. Da allora fino al giorno d'oggi, nella progettazione dei calcolatori elettronici e dei microprocessori, ci si è basati sui principi abbozzati in questo documento.
Mai come verso la metà di questo periodo, durante la Seconda Guerra Mondiale, ci

fu un pari grado di collaborazione tra scienziati, ingegneri, uomini politici e militari, che si erano uniti nel tentativo di sconfiggere le Potenze dell'Asse facendo ricorso all'intelligenza, dato che la sola forza non sarebbe bastata. In Gran Bretagna il nucleo di questa collaborazione era un istituto segretissimo a Bletchley Park, dove un piccolo gruppo capeggiato da Max Neumann, a cui appartenevano anche Donald Michie e Tommy Flowers, progettarono una serie di elaboratori per decifrare i codici segreti dell'esercito tedesco. Questi, nel 1943, portarono alla realizzazione delle macchine elettroniche Colossus. Basate su principi di criptoanalisi messi a punto da Turing, permisero di decifrare quei codici dell'Alto Comando tedesco prodotti dalle macchine Enigma che fino ad allora erano considerati sicuri. Ora tutti riconoscono che la parte che ebbero le varie macchine del gruppo di Bletchley fu molto importante, se non decisiva, nella Battaglia dell'Atlantico e, forse, nell'intero conflitto. I dettagli dell'attività svolta a Bletchley Park sono rimasti segreti; comunque molti fra coloro che vi hanno lavorato hanno fatto carriera nel settore informatico e si sono occupati anche di intelligenza artificiale.
Dopo la guerra, le potenzialità degli elaboratori elettronici nel settore militare, industriale e commerciale divennero evidenti. Nei quaranta anni che seguirono si verificò una crescita esponenziale nel settore informatico, dato che si cominciò a comprendere la portata di queste potenzialità. L'andamento dei progressi compiuti non è stato sempre regolare, e, di tanto in tanto, si possono notare alcuni segni di saturazione di determinati mercati; in generale, però, si continua a progredire. La convergenza dei computer, delle telecomunicazioni e dei sistemi di controllo elettronici ha dato origine alla tecnologia dell'informazione, che probabilmente, entro la fine di questo secolo, diventerà la più grossa industria nel mondo occidentale. In generale, si possono identificare cinque linee di sviluppo:

- teoria fondamentale,
- progettazione dell'hardware,
- programmazione del software,
- applicazioni,
- intelligenza artificiale.

Mentre ci sono sempre state continue interferenze tra le prime quattro, la quinta, fino a poco tempo fa, è sempre rimasta un po' isolata rispetto alla corrente principale. Comunque, questa situazione sta volgendo alla fine grazie alle pressioni sorte in

seguito allo sviluppo della quinta generazione di computer.

1.2 Il concetto di computer: l'opera di Turing e Church

Per avere nozioni chiare sulla quinta generazione di computer è necessario approfondire il concetto di computer. Volendo dare una spiegazione superficiale, un computer è una macchina digitale elettronica che elabora informazioni ed è controllata da un programma memorizzato. Ma questo concetto superficiale che, in un futuro non lontano, potrebbe apparire piuttosto limitato, è il prodotto di pensieri molto più profondi. La frase chiave, "elaborazione di informazioni" venne coniata da Marshall MacLuhan quando l'IBM chiese di spiegare succintamente che cosa fossero i computer.
Fu all'inizio di questo secolo, quando ci si rese conto che la matematica può essere considerata una scienza completamente astratta, che si scoprì la chiave per capire chiaramente il concetto di computer. La matematica riguarda simboli, relazioni tra simboli, operazioni su simboli e le proprietà di questi simboli, relazioni e operazioni. Il fatto che molti di questi simboli, come quelli che rappresentano i numeri, trovino una realizzazione concreta, non altera affatto le loro proprietà matematiche astratte. Cominciando da pochi assiomi fondamentali e precise definizioni dell'oggetto, è possibile tessere tutta l'intricata tela che costituisce la matematica moderna per mezzo di teoremi e corollari, deduzioni e induzioni, senza ricorrere alla realtà esterna.
Nel tentativo di "circoscrivere" gli studi sulla matematica astratta, David Hilbert ad un congresso internazionale nel 1928, pose tre quesiti fondamentali. Primo: la matematica è completa, nel senso che è sempre possibile dimostrare se un'affermazione matematica è vera o falsa? Secondo: la matematica è coerente, nel senso che, in una dimostrazione, nessun passo corretto può portare a una conlusione non corretta? Terzo, la matematica è decidibile, ossia esiste un metodo ben preciso che, in linea di principio, possa essere applicato a qualsiasi enunciato matematico e che stabilisca se l'affermazione è vera? L'intuizione di Hilbert lo portò a pensare che tutte tre le risposte a queste domande fossero affermative. Otto anni dopo, però si dimostrò che aveva sbagliato completamente; fu così che il suo concetto di "metodo preciso" sfociò nell'idea, ancora teorica, di un calcolatore universale.
Allo stesso convegno nel 1928, Kurt Gödel riuscì a dimostrare che l'aritmetica,

all'interno del suo sistema assiomatico, non può essere dimostrata coerente e che non è neppure completa (Davis, 1965). Max Newman, che vi aveva assistito, tenne delle lezioni a Cambridge sui quesiti sollevati da Hilbert. Uno dei suoi studenti, Alan Turing, si soffermò su un'espressione di Newman, "un processo meccanico", e, nel 1936, presentò quale argomento della sua tesi un notevole scritto intitolato "On Computable Numbers, with an Application to the Entscheidungsproblem" (Turing, 1936).

Turing affermò che la terza domanda di Hilbert — l'*Entscheidungsproblem* — può essere risolta con l'aiuto di una macchina, o almeno con il concetto astratto di una macchina. Una macchina di Turing (come oggi è noto) è in grado di calcolare il valore di qualsiasi numero che sia specificato da un procedimento preciso (ciò che noi chiameremmo un algoritmo) — da qui l'idea di numeri computabili. Per esempio, il numero *pi*, non essendo razionale, ha uno sviluppo decimale infinito, ma il suo valore può essere espresso per mezzo di un algoritmo e quindi essere calcolato. Così, dandole il tempo sufficiente, una macchina di Turing può calcolare il valore di pi con qualsiasi precisione richiesta. Pur non dimostrando formalmente la capacità di questa macchina, Turing presentò una serie di convincenti motivazioni che fecero sì che la tua tesi sia ora accettata e considerata un assioma della teoria del calcolo. Egli si servì del suo concetto di macchina per dimostrare che esistono alcuni numeri non computabili con un metodo ben definito, e che pertanto la risposta alla terza domanda di Hilbert è negativa.

Nello stesso periodo, quasi contemporaneamente, negli USA Alonzo Church arrivò a quella stessa conclusione (Church, 1936). Church si servì di una notazione formale che chiamò "lambda-calcolo", per tradurre tutte le formule aritmetiche in una forma standard. In questo modo dimostrare un teorema diventava un problema di trasformazione di una stringa di simboli del lambda-calcolo in un'altra, secondo un insieme di regole formali. Non esiste, comunque, alcuna formula del lambda-calcolo per determinare, in termini generali, se sia possibile tradurre una stringa di simboli in un'altra. Ancora una volta ci sono dei casi in cui non può essere applicato un "metodo preciso". Un corollario dei risultati di Church fa eco alla tesi di Turing: l'insieme dei numeri computabili è costituito esattamente da quei numeri che hanno una formula corrispondente nel lambda-calcolo.

L'opera di Turing e Church ha influito notevolmente sullo sviluppo dei computer. Innanzitutto, un numero non è che una stringa di simboli che possono essere interpretati in qualsiasi modo. Conseguenza alquanto banale è che qualsiasi base

numerica può essere utilizzata per effettuare un calcolo. Questi sono i ragionamenti formali impliciti nell'idea secondo cui i dati di un computer sono una sequenza di simboli manipolati dalla macchina che dovrebbero essere interpretati da un utente perché possano fornire un'informazione. Ancor più importante è che una stringa di simboli può essere interpretata come un'istruzione. In secondo luogo, una macchina di Turing è universale, dato che è in grado di risolvere qualsiasi problema esprimibile per mezzo di un algoritmo (in altre parole, può computare qualsiasi numero computabile). In ultima analisi, una macchina di Turing è, in linea di principio, molto semplice. Ha un'unico nastro per l'input, l'output e la memoria; è controllata da "tabelle di comportamento" (programmi) e può trovarsi in qualsiasi momento in uno qualunque di un certo numero di stati. Ogni operazione comporta la manipolazione di un singolo simbolo sul nastro e talvolta lo scorrimento del nastro, nell'una o nell'altra direzione. La costruzione di un calcolatore di uso generale che realizzi il concetto della macchina di Turing non dovrebbe, in pratica, essere impossibile.

I successivi sviluppi della teoria della computazione hanno seguito le linee stabilite da Turing e Church — rispettivamente l'approccio della macchina astratta e quello della programmazione funzionale (Brady, 1977). Il concetto della macchina di Turing è stato chiarito e perfezionato, e le sue caratteristiche sono state esaminate molto dettagliatamente. La programmazione funzionale ha portato allo sviluppo di linguaggi di programmazione quali il Lisp (vedi paragrafo 7.1), il linguaggio di quasi tutte le applicazioni dell'intelligenza artificiale, ed all'idea che un programma può essere considerato una successione di enunciati matematici la cui correttezza può essere dimostrata o confutata per mezzo di tecniche formali matematiche.

1.3 Il calcolatore elettronico: l'opera di Boole e Shannon

Con l'opera di Turing e Church si è stabilito che un computer è essenzialmente una macchina che manipola simboli ma lascia qualsiasi interpretazione dei simboli al programmatore o all'utente. Le operazioni primitive effettuate da un computer appartengono a un livello inferiore a quello della matematica, il regno della logica. A questo livello i simboli sono più accessibili sotto forma di cifre binarie — bit — che possono rappresentare lettere, colori su uno schermo grafico, o componenti di un suono. I circuiti di commutazione elettronica che eseguono queste operazioni

primitive corrispondono esattamente alle cifre binarie 0 ed 1: possono trovarsi solo in uno di due stati.
Per individuare l'origine degli studi che ci hanno condotto a questa situazione si può risalire fino alla logica di Aristotele (384 a.C-322 a.C). Aristotele stabilì le regole della logica formale, in base a cui la verità o falsità di un'affermazione può essere dedotta attraverso un procedimento ben definito. In ordine d'importanza, il secondo passo avanti fu compiuto da George Boole (1815-1864), che in due delle sue opere (Boole, 1848 e 1854) dimostrò che la logica aristotelica può essere espressa con una notazione algebrica. Il legame tra logica ed elettronica venne stabilito da Claude Shannon (nato nel 1916) che dimostrò come le operazioni elementari booleane possano essere rappresentate per mezzo di circuiti di commutazione elettrica e come le combinazioni di questi circuiti possano rappresentare complesse operazioni logiche e aritmetiche (Shannon, 1938). Ancor più importante è il fatto che Shannon mostrò come l'algebra booleana possa essere usata per semplificare i circuiti di commutazione e che le proprietà di tali circuiti possono essere stabilite per mezzo di dimostrazioni formali.

1.4 Il calcolatore a programma memorizzato: l'opera di Von Neumann

Tra il 1936 e il 1946, l'ultimo anello della catena di sviluppi attraverso cui si arrivò al concetto di computer moderno venne fornito da John Von Neumann (1903-1957) e dai suoi colleghi negli istituti di ricerca americani, durante la guerra. Avendo già partecipato in qualità di consulente alla messa a punto del progetto e della costruzione di uno dei primi elaboratori elettronici (il computer Eniac, il primo e ultimo elaboratore elettronico a base dieci), Von Neumann fu in grado di individuare i requisiti della struttura di un calcolatore elettronico universale (Von Neumann, 1945; Burks, Goldstine e Von Neumann, 1946). Questo elaboratore doveva operare su dati e istruzioni in codice binario, presenti nella sua memoria. Erano sufficienti pochi registri per conservare le istruzioni del programma in esecuzione e i dati in elaborazione. La macchina, inoltre, doveva eseguire cicli ripetuti di passi per prelevare ed eseguire in sequenza le istruzioni di un programma.
Questa descrizione corrisponde, più o meno, a tutti i calcolatori elettronici costruiti dal 1946. I requisiti delle operazioni seriali sono tuttora validi per la maggior parte delle unità di elaborazione singole, ma la maggior parte dei calcolatori elettronici

moderni contiene molte unità di elaborazione che, in certa misura, lavorano in parallelo. Uno dei principali requisiti dell'hardware dei computer della quinta generazione è un più potente funzionamento in parallelo.
L'idea di memorizzare insieme programmi e dati comporta delle profonde conseguenze. Da un punto di vista pratico, permette di risparmiare spazio nella memoria consentendo una partizione più flessibile dell'area memoria tra dati e istruzioni. Ciò significa che i meccanismi per prelevare dati e istruzioni dalla memoria sono gli stessi. Cosa ancor più fondamentale, significa che i simboli possono essere trattati come dati in un contesto e come istruzioni in un altro: ad esempio, i codici sorgente di un linguaggio di alto livello sono, per il programmatore, istruzioni, ma, per il compilatore, dati. Fondamentale conseguenza di ciò è che i programmi possono, in teoria, modificare se stessi alla luce delle loro stesse operazioni. A Von Neumann e agli altri studiosi all'inizio degli anni cinquanta, questa era sembrata una linea di sviluppo molto promettente; purtroppo, però, non ha portato a grossi progressi. Attualmente i programmi in grado di modificare se stessi non sono visti di buon occhio, ma i nuovi sviluppi sull'intelligenza artificale potrebbero capovolgere questa situazione.

1.5 Implementazione elettronica: valvole, transistor e microchip

Il primo elaboratore realizzato sulla linea di Von Neumann fu il Manchester University Mark 1, un piccolo prototipo che venne reso funzionante nel 1948. Ben presto ne furono creati degli altri e fu solo tre anni dopo che entrò in funzione il primo elaboratore elettronico commerciale (Univac). Questi colossi di valvole, poveri in potenza di elaborazione se confrontati con i moderni minielaboratori, furono rimpiazzati, verso la fine degli anni cinquanta, dalle versioni transistorizzate. L'invenzione del transistor, nel 1948, fu la scoperta che ha maggiormente contribuito a rendere i computer ampiamente accessibili. Infatti, ben presto si scoprì che, oltre ad essere piccoli, economici e molto affidabili, i transistor possono essere combinati tra loro e con altri elementi di un circuito per formare un unico circuito integrato — il chip onnipresente. La serie di computer IBM 360 comparve a metà degli anni sessanta e contribuì a consolidare il dominio dell'IBM sul mercato informatico mondiale. Nel 1972 si raggiunse il massimo grado di integrazione — un'intera unità di elaborazione su un singolo chip. Alla fine degli anni settanta i

minicomputer da tavolo erano ormai presenti ovunque e nel 1982 venne prodotto il milionesimo calcolatore elettronico digitale a programma memorizzato.
Le quattro generazioni di implementazione dell'hardware (valvole, transistor, circuiti integrati e microprocessori) hanno visto una crescita esplosiva del settore informatico e il passaggio a un diverso utilizzo del computer che, dai segretissimi istituti militari, è passato al tavolo da cucina. Ciononostante, la maggior parte dei pionieri del periodo 1936-1946 si troverebbero a loro agio davanti a un personal e si accorgerebbero immediatamente che le sue origini risalgono ai tempi della Seconda Guerra Mondiale.

1.6 Sviluppi del software

Con l'aumento della potenza e della complessità della struttura hardware dei computer, diventò necessario introdurre sempre più software quale interfaccia tra hardware e utente. È normale che gli unici a capire completamente le particolari operazioni della struttura hardware di un elaboratore siano gli ingegneri diretti responsabili della sua progettazione e della sua costruzione. Dato che questo limitava l'uso di questi computer, vennero introdotti diversi livelli di software ognuno dei quali, presentando una visione semplificata del computer al software e agli utenti esterni, li rendeva più facilmente accessibili ai programmatori e agli utenti.
Lo strato più interno, che oggi chiamiamo sistema operativo, sostituisce la vera struttura hardware con una macchina virtuale sulla quale, ad esempio, i dati vengono trasferiti avanti e indietro dalle unità periferiche sotto forma di entità logiche (archivi, record, ecc.) anziché di entità fisiche. Il livello successivo, presente solo durante lo sviluppo del software, è costituito dal traduttore dei vari linguaggi che permette ai programmatori di utilizzare linguaggi di alto livello rivolti all'utente anziché linguaggi macchina. Per questi traduttori di linguaggi le istruzioni di un linguaggio di alto livello sono dati che essi trasformano in un insieme di istruzioni macchina equivalenti che vengono poi controllate dal sistema operativo.
L'interfaccia più esterna, e per molti aspetti la più importante, è quella resa visibile dai programmi applicativi. Questa è l'interfaccia utente, che interagisce tra il computer e la persona che lo utilizza. Un programma applicativo può essere visto come un modo per trasformare un calcolatore di uso generale in un calcolatore dedicato a compiti particolari, in cui l'interfaccia utente costituisce il mezzo di

interazione con la macchina specializzata.
Fino a poco tempo fa, l'interfaccia utente veniva considerata il punto debole di quasi tutti i sistemi computerizzati. In molti casi essa veniva "innestata" sul software come se fosse un'aggiunta, senza tener conto delle necessità, delle conoscenze o dei processi mentali degli utenti. L'inadeguatezza dell'interfaccia utente è sempre stata uno dei fattori che hanno maggiormente ostacolato l'accessibilità dei computer. Probabilmente, questa situazione cambierà grazie a una miglior comprensione della psicologia nell'interazione uomo-macchina e ai progressi compiuti nella progettazione delle interfacce utente, risultato di studi svolti sull'intelligenza artificiale. La strutturazione delle interfacce utente (capitolo 9) è uno dei punti fondamentali nello sviluppo della quinta generazione di computer.

1.7 L'approccio sistemico

È indubbio che i computer siano quanto di più complesso si sia mai prodotto. Il fatto che comincino a essere utilizzati da gran parte della popolazione di molti paesi industriali, compresa la maggior parte degli scolari, è una situazione senza precedenti nella storia. Per destreggiarsi in questa complessità, è necessario ricorrere a svariati strumenti concettuali, quali quello dei diversi strati di software e delle interfacce sopra descritte. Quello maggiormente usato, anche se, forse, spesso poco conosciuto, è il concetto di sistema. Molti professionisti dell'informatica ricorrono alla frase "il sistema" in mancanza di qualcosa di più preciso, creando così gran confusione nella mente dei loro interlocutori.
Quello di sistema è, però, un concetto ben preciso (Beishon e Peters, 1972), necessario per ridurre la complessità dell'hardware e del software dei computer moderni portandola a proporzioni gestibili. In generale, un sistema può essere considerato come un insieme di elementi correlati, che assieme raggiungono (o cercano di raggiungere) scopi e obiettivi ben precisi. Un sistema può avere dei sottosistemi, che comprendono sottoinsiemi dei suoi elementi e che raggiungono sottoinsiemi dei suoi scopi e dei suoi obiettivi. A sua volta, ogni sistema può essere un sottosistema di un sistema più grosso. Un sistema ha confini ben precisi, attraverso i quali interagisce con il suo ambiente. Questo corrisponde proprio all'idea di interfaccia di cui si è parlato sopra.
L'applicazione del concetto di sistema a un computer semplifica molto i problemi.

Per esempio, un sistema di elaborazione dati può comprendere solo l'hardware o il software o l'insieme dei due. È abbastanza corretto descrivere un sistema in base ai suoi fini e obiettivi, o il flusso d'informazione in base ai suoi confini, senza specificare assolutamente i componenti del sistema. Questo porta all'idea di modulo, ossia di un sistema delimitato in modo ben preciso e dotato di interfacce molto semplici, che può essere "staccato" e sostituito da un altro modulo con le stesse interfacce e funzioni, senza alterare assolutamente l'ambiente del modulo. Un chip è un esempio banale di modulo hardware. Attualmente si sta cercando in tutti i modi di scrivere software costituito interamente da moduli che siano staccabili e sostituibili nello stesso modo.
Buona parte dell'attività in campo informatico è svolta secondo questa filosofia sistemica. Le applicazioni dell'elaborazione, i componenti dei computer, i vari strati di software vengono descritti in base ai loro scopi e agli obiettivi, ai loro confini e alle loro interfacce, e i dettagli degli elementi che li compongono vengono dati solo in caso di estrema necessità. Per evitare qualsiasi malinteso conseguente al notevole aumento della complessità dei sistemi di elaborazione, con l'avvento della quinta generazione di computer questo approccio sistemico dovrà diventare più rigoroso.

1.8 L'informatica all'inizio degli anni ottanta

Così siamo arrivati dall'*Entschiedungsproblem* — il problema di un metodo ben definito — alla proliferazione dei computer all'inizio degli anni ottanta. I concetti comuni di computer quale elaboratore di informazioni, di dati quali stringhe di simboli gestiti da un computer e interpretati dall'utente, e di programmi quali trasformatori di un computer da elaboratore di uso generale a elaboratore dedicato, sono frutto di alcune tra le menti più brillanti del ventesimo secolo. La maggior parte degli utenti dei computer conoscono abbastanza bene le potenzialità e i limiti di queste macchine. Esse, infatti, sono in grado di ordinare, selezionare, confrontare e combinare i dati, di effettuare operazioni di calcolo e persino di prendere decisioni basate sui dati e possono fare tutto questo a velocità irraggiungibili dall'uomo. Ma i computer non sono in grado di capire i dati che stanno elaborando, non possono prendere l'iniziativa da soli e neppure fornire una spiegazione se i dati sono poco chiari, incompleti o contradditori. Non possono inoltre operare con un linguaggio

naturale e per quanto producano splendide immagini grafiche, non sanno interpretare le informazioni in forma visiva. La loro abilità non è che il prodotto di una serie di istruzioni che controllano sia loro stessi che i dati su cui operano queste istruzioni.
I computer sono diventati gli strumenti standard delle scienze "esatte" — fisica, chimica e, in certa misura, biologia. Essi sono indispensabili in tutti i settori dell'ingegneria e in scienze quali l'aereonautica e l'astronautica dove è proprio grazie a loro che si sono potuti realizzare i progressi compiuti negli ultimi quarant'anni. Tuttavia, i computer non sono ancora altrettanto importanti in discipline quali la sociologia, l'economia e, soprattutto, la medicina dove i concetti base, non essendo del tutto precisi, non sono, pertanto, facilmente quantificabili. In questi settori è ancora il professionista specializzato ed esperto a prendere le ultime decisioni.
Questa situazione cambierà notevolmente se si realizzeranno alcune delle aspirazioni della quinta generazione di computer.

2

Intelligenza artificiale

> Sappiamo perfettamente che, negli esseri umani, esiste uno spazio tra il comportamento totalmente deterministico e il comportamento totalmente caotico. Questo è lo spazio dell'intelligenza. Il problema è semplicemente che, nell'intero corso della loro storia, le macchine non hanno mai occupato questo spazio. Forse stiamo vivendo sul fronte dell'onda della rivoluzione culturale più importante della storia umana, ma le credenze mutano lentamente e dobbiamo ancora renderci conto del fatto che "intelligenza" non significa più, soltanto, intelligenza umana.
> (Harold Cohen, artista informatico, nella Tate Gallery, 1983).

Fin da quando Charles Babbage concepì l'idea di un calcolatore automatico, controllato da una sequenza di istruzioni, si è sempre discusso molto per stabilire in quale misura l'intelligenza venga impartita a un calcolatore dai suoi creatori umani e dai programmatori. Sia Turing che Von Neumann, negli ultimi anni della loro carriera, hanno dedicato molto tempo al problema dell'intelligenza delle macchine e un piccolo, ma spesso agguerrito gruppo di studiosi è riuscito a mantenere vivo il problema fino al giorno d'oggi. I punti centrali del dibattito sono:

Quanto è intelligente un computer?

e

Quali sono i fini pratici per i quali si può usare l'intelligenza di un computer?

Questi problemi possono essere esaminati da due punti di vista. In primo luogo, accettando il concetto secondo cui un computer è un manipolatore automatico di simboli che non può andare oltre alle capacità attribuitegli dalla sua struttura hardware e software, quanta intelligenza può essere trasferita a una macchina di questo tipo? Fino a che punto l'intelligenza umana è capace di esprimersi in termini di manipolazione di simboli? Il secondo approccio è quello di riesaminare il concetto fondamentale di calcolatore che abbiamo appreso dall'opera di Babbage, Turing, Von Neumann e altri, per vedere se sia possibile rimuovere o ridurre alcuni dei limiti che esso comporta. Fino a ora, la maggior parte delle ricerche sull'intelligenza artificiale ha optato per il primo approccio; molta attività della quinta generazione rientra invece nella seconda categoria.
Alla base di questi quesiti sta una difficoltà ancor più fondamentale: la nozione di intelligenza umana non è definita in modo abbastanza chiaro da permettere che i concetti di intelligenza artificiale possano basarsi su di essa. Per questo e per altri motivi, non esiste una definizione universalmente accettata del termine "intelligenza artificiale" (Boden, 1977) e le due grandi linee di ricerca — sulle teorie fondamentali che stanno alla base di ampie classi di applicazioni, e sui metodi euristici che si applicano in particolari situazioni — si sono talvolta trovate in disaccordo. Anni di intense ricerche hanno presentato allettanti prospettive per il futuro, ma pochissimi progressi concreti. Per esempio, per quanto oggi i calcolatori sappiano effettivamente giocare a scacchi da gran maestri, saper identificare i pezzi gettati alla rinfusa in una scatola e sistemarli sulla scacchiera nella posizione corretta è decisamente al di là delle possibilità anche del più sofisticato robot a controllo computerizzato.
In Gran Bretagna, le ricerche sull'intelligenza artificiale hanno subito un grave arresto in seguito alla pubblicazione del Lighthill Report (Lighthill, 1972), in cui si arrivava alla conclusione che la maggior parte delle applicazioni dell'intelligenza artificiale sono così caratterizzate dalla varietà delle situazioni possibili che i calcolatori, occupandosi di esse, finiranno per rimanere intrappolati in una "esplosione combinatoria". Questo significa che, nei pochi casi in cui (come nel gioco degli scacchi) l'intelligenza artificiale ha fatto dei progressi, il numero di

possibili combinazioni di eventi, per quanto molto alto, è gestibile soltanto attenendosi scrupolosamente alle regole e strategie stabilite. Comunque, in quasi tutte le situazioni della vita reale, il numero di possibilità è troppo alto perché ciascuna possa essere considerata individualmente. Ad esempio, è piuttosto facile programmare un camputer perché sappia distinguere fra loro forme semplici quali, ad esempio, cubi e piramidi, ma renderlo capace di distiguere delle forme più complesse quali, ad esempio, i diversi tipi di dadi e bulloni o i volti della gente, è quasi impossibile. Gran parte dell'attività nel settore dell'intelligenza artificiale è volta soprattutto allo sviluppo di strategie di ricerca per ridurre il numero di possibilità in qualsiasi situazione particolare, ma questo non è sufficiente a ridurre le conseguenze del Lighthill Report. Per quanto la comparsa dei computer della quinta generazione, risultato di sviluppi sull'intelligenza artificiale, abbia segnato la ripresa delle ricerche sull'intelligenza artificiale in Gran Bretagna, il numero di ricercatori esperti nel settore rimane piuttosto esiguo.
Questo capitolo analizza alcune nozioni dell'intelligenza umana e presenta una definizione pratica del concetto di intelligenza artificiale. Esamina i modi di rappresentare la conoscenza su un computer e passa in rassegna i vari campi in cui le ricerche sull'intelligenza artificiale stanno facendo dei progressi.

2.1 Nozioni di intelligenza

Tutti siamo d'accordo sul fatto che alcune persone sono più intelligenti di altre, che la maggior parte delle persone è più intelligente degli scimpanzé e che gli scimpanzé sono più intelligenti delle tenie. Inoltre, la maggior parte della gente riconosce che anche le tenie possiedono un certo grado, per quanto minimo, di intelligenza. In modo analogo, la maggior parte delle persone ritiene che un *word processor* sia più intelligente di una comune macchina per scrivere. Per quanto, in questo modo, si possa misurare in modo approssimativo il grado relativo di intelligenza, le qualità che distinguono l'intelligenza rimangono di difficile comprensione. L'intelligenza è legata alla capacità di riconoscere le forme, trarre conclusioni logiche, scomporre sistemi complessi in elementi semplici e risolvere contraddizioni; ma è più di tutto questo. Essa contiene quell'indefinibile "illuminazione" che permette di intuire nuovi concetti, di formulare nuove teorie e di giungere a nuove conoscenze.
L'intelligenza può essere collocata in una gerarchia in cui il livello più basso

corrisponde all'informazione. L'informazione consiste dei fatti (o di ipotetici fatti) che costituiscono la materia prima dei livelli superiori. L'informazione è di facile acquisizione — può essere scritta, immessa in un sistema computerizzato o imparata a memoria. Il livello superiore è costituito dalla zona della conoscenza: associazione tra fatti, formule matematiche ecc. La conoscenza non è di facile acquisizione come l'informazione — richiede un processo di apprendimento complesso e ancora poco chiaro. Al di sopra di questa si trova l'intelligenza che opera sull'informazione e sulla conoscenza. Il grado di intelligenza è innato, e viene realizzato in misura maggiore o minore attraverso la conoscenza e l'esperienza. Al vertice della gerarchia sta la saggezza, che spesso ha connotazioni mistiche o religiose, e la cui definizione è ancora più complessa di quella dell'intelligenza.
Questa gerarchia può essere esaminata anche dal punto di vista del linguaggio. L'informazione può essere rappresentata facilmente per mezzo di parole, numeri o altri simboli. La conoscenza viene generalmente espressa in forma linguistica o matematica e la sua rappresentazione costituisce una delle imprese più ardue per quanti si occupano dei computer della quinta generazione. L'intelligenza è al di sopra del limite massimo del linguaggio: infatti, si possono descrivere modelli di strutture e ragionamenti deduttivi e si possono enunciare alcuni principi generali, ma l'"illuminazione" creativa dell'intelligenza va oltre le capacità espressive del linguaggio. La saggezza sta quasi completamente oltre la portata del linguaggio: a questo livello, i tentativi di sconfinare nel mondo delle parole trovano conferma in enigmi del tipo "Qual è il suono di una mano che applaude?"
Un'ulteriore problema sorge quando si usa il linguaggio per descrivere aspetti dell'intelligenza, sia naturali che artificiali. Questo avviene perché i termini utilizzati in tali descrizioni sono "carichi" di vari presupposti filosofici, psicologici e persino politici (Boden, 1977). La stessa parola "intelligenza" ne è un buon esempio. Per alcuni, di ideologia comportamentista, il termine ha un significato stretto e preciso ed è in qualche modo quantificabile. Secondo altri, di inclinazione più umanistica, il termine ha una significato molto profondo. Il problema dell'uso delle parole senza una definizione precisa e universalmente accettata è infinito — le parole vengono definite per mezzo di altre parole, e utilizzate con propositi che sono sempre discutibili, ecc.
Da studi in campo psicologico sta diventando sempre più chiaro che ciò che intendiamo per mente cosciente, razionale non è che la punta dell'iceberg dell'intelligenza. Al di sotto vi sono strati di subconscio, che operano in modo solo vaga-

mente noto e che solo occasionalmente lasciano emergere sulla superficie della consapevolezza immagini coerenti. Ciò nonostante, questi strati di subconscio sono parte integrante dell'intero processo mentale, aspetto spesso ignorato nelle opere sull'intelligenza artificiale. Uno dei personaggi più intelligenti vissuti nel nostro secolo, l'artista Salvator Dali, ha chiarito che la visione celata nelle sue opere non è che l'insieme delle immagini presentategli dalla sua mente subcosciente.

La fisiologia del cervello e del sistema nervoso è una zona grigia in senso lato. Stiamo incominciando a capire come i segnali vengano trasmessi attraverso le cellule nervose e, grazie soprattutto agli studi effettuati su vittime di incidenti, stiamo incominciando a capire quali siano le zone del cervello responsabili dei processi mentali. Sappiamo che l'"elaborazione" svolta all'interno del cervello è un'elaborazione in parallelo ad altissimo livello e che non esiste una chiara distinzione in termini "hardware" tra "elaborazione" e "memoria".

Sappiamo che, per quanto le cellule che si trovano in parti diverse del cervello siano a loro volta diverse, in determinate circostanze, possono svolgere le funzioni di altre. In altre parole, il cervello ammette un buon margine di errore. Comunque, abbiamo solo una vaga idea di quale sia il meccanismo di base del pensiero e della memoria e, a quanto pare, siamo ancora lontani dalla sua scoperta. Di tanto in tanto vengono fatte delle analogie tra l'attività dei circuiti di un computer e quella del cervello. Talvolta si confrontano quantitativamente la potenza di elaborazione, la velocità di operazione o la capacità di memoria di un computer e il cervello umano. Nessuno di questi confronti potrebbe reggere a un esame attento — si confrontano infatti elaborazioni di informazioni digitali con materiali semiconduttori al silicio con un processo biologico che è di gran lunga più complesso e sul quale sappiamo molto poco.

Da un punto di vista filosofico, negli ultimi secoli, la teoria della conoscenza ha subito tanti cambiamenti che si è giunti al punto in cui predomina la convinzione che nulla sia noto con assoluta certezza:

> La conoscenza, e soprattutto la nostra conoscenza scientifica, procede per mezzo di ingiustificate (e ingiustificabili) previsioni, per ipotesi, soluzioni sperimentali ai nostri problemi, congetture. Queste congetture sono controllate dalla critica; ossia da tentativi di confutazioni attraverso precise prove di verifica. Esse possono superare queste prove, ma non possono mai essere spiegate con certezza: non possono né essere dichiarate certamente

vere, né almeno "probabili" (in termini di calcolo delle probabilità). (Popper, 1963)

Ogni sistema che automatizzi la conoscenza è soggetto a queste limitazioni.
La conclusione che si deve trarre da queste varie linee di pensiero è che non c'è una ben definita linea di demarcazione dell'intelligenza "naturale" sulla quale misurare l'intelligenza artificiale, e che gli sviluppi sull'intelligenza artificiale si sono basati su principi teorici che sono ancora motivo di acceso dibattito. Questi sono stati alcuni dei più seri problemi nello sviluppo dell'intelligenza artificiale, che, come vedremo in seguito, sono stati affrontati o evitati in molti modi ingegnosi.

2.2 L'intelligenza del calcolatore

Non appena i primissimi computer vennero denominati "cervelli elettronici" (Hodges, 1983) si comiciò a discutere per stabilire fino a che punto fossero effettivamente intelligenti.
Vennero fatti vari tentativi per trovare strette analogie tra l'elaboratore elettronico e il sistema nervoso, come nell'ultima opera di Von Neumann (Von Neumann, 1951) e del neurologo W. Ross Ashby (Ashby, 1952) che diedero adito a intense discussioni e speculazioni ma non determinarono notevoli progressi.
Nel 1950, Turing si accostò al problema dell'intelligenza del computer con lo stesso approccio già adottato nel 1936 per il concetto generale di elaboratore elettronico. Dopo aver assistito, nel 1949 presso l'Università di Manchester, a una discussione su "La mente e l'elaboratore elettronico", presentò uno scritto (Turing, 1950) in cui esprimeva le proprie idee sull'argomento. In questa sua opera è contenuto ciò che oggi è ampiamente riconosciuto essere un test pratico per l'intelligenza artificiale: se in una stanza vi sono due terminali identici, uno collegato a un computer, l'altro gestito, in qualche modo, a distanza da una persona, e se qualcuno che sta utilizzando i due terminali è incapace di decidere quale sia collegato al computer e quale sia controllato dall'essere umano, allora si può attribuire intelligenza al computer.
Il test di Turing eludeva tutti i problemi con le nozioni filosofiche, psicologiche e fisiologiche di intelligenza di cui abbiamo parlato nel paragrafo precedente. Secondo questo test non si può dare una definizione di intelligenza di per se stessa,

ma si può riconoscere il comportamento intelligente. Tuttavia, per quanto esso sia stato criticato da più punti di vista (è del tutto soggettivo, non dà alcuna indicazione su come potrebbe essere costruito un sistema computerizzato, effettua confronti fra cose cui non è neppure applicabile la stessa unità di misura ecc.), è ampiamente accettato e fornisce una delle definizioni di intelligenza artificiale meno controverse: "L'intelligenza artificiale è la scienza che fa fare alle macchine cose che, se fatte dall'uomo, richiederebbero intelligenza" (Minsky, 1968). Un vantaggio di questa definizione è che ammette diversi livelli di intelligenza e misura il grado di intelligenza in base all'avvenuto adempimento del compito. Un computer che calcola uno stipendio dà prova di una certa quantità di intelligenza, e così farebbe un computer se dovesse dedurre il processo fondamentale della crescita del cancro. È proprio questo crescente avanzamento sul fronte dell'intelligenza delle macchine che sta alla base del concetto della quinta generazione di computer.
Nessun sistema computerizzato è mai stato prossimo al superamento del test di Turing in termini generali. Ciò nonostante, negli anni ottanta un giocatore di scacchi dovrebbe essere veramente molto bravo per riconoscere se stia giocando contro un avversario umano o contro un computer. Allo stesso modo, la maggior parte degli automobilisti non sa quali parti della loro automobile siano state montate da robot e quali da operai. I progressi compiuti in questo e in altri campi dell'intelligenza artificiale verranno presentati nei prossimi paragrafi di questo capitolo.

2.3 Informazione e conoscenza

L'elaborazione tradizionale è basata sull'informazione; l'intelligenza artificiale è basata sulla conoscenza. Come abbiamo visto nei paragrafi precedenti, per elaborare la conoscenza è necessario che il numero di possibilità da considerare in ogni particolare situazione rimanga entro limiti gestibili. Pertanto, un problema centrale dell'intelligenza artificiale è riuscire a rappresentare la conoscenza su un computer in modo adeguato. Da un lato, la rappresentazione deve essere sufficientemente "ricca" perché sia effettivamente utilizzabile. Allo stesso tempo deve essere abbastanza semplice da formare la base di strategie automatiche per trarre conclusioni valide e utili partendo da certe conoscenze, all'interno dei limiti dell'hardware e del software dell'unità di elaborazione principale.
La linea generale dell'approccio più riuscito consiste nel rappresentare un sistema

basato sulla conoscenza per mezzo di tre livelli (Alty e Coombs, 1984). Il livello più basso è quello delle associazioni tra gli oggetti. Ad esempio, "la temperatura alta è sintomo di influenza" o, più formalmente:

sintomo-di (temperatura alta, influenza).

In un sistema basato sulla conoscenza, associazioni espresse in questa forma sono enunciati o proposizioni e le relazioni (come "sintomo di") costituiscono i predicati. Gli oggetti all'interno di una proposizione possono essere costanti (ad esempio "alta temperatura", "influenza"), o variabili, per comodità denotate da lettere minuscole come in:

sintomo-di (*x*, *y*)

che sta a indicare che x è un sintomo di y.

Le proposizioni possono correlare più di due oggetti o possono essere affermazioni riguardanti un singolo oggetto. Ad esempio:

sintomo-di (stomaco dilatato, malnutrizione, bambino)

sta a significare che lo stomaco dilatato nel bambino è sintomo di malnutrizione e:

malattia (influenza)

indica che l'influenza è una malattia.
Le proposizioni possono esprimere anche una probabilità o un fattore ponderale utilizzando una scala adatta. Ad esempio:

Sintomo-di (temperatura alta, influenza, 8)

significa che la temperatura alta è sintomo d'influenza con un fattore ponderale di 8 in una scala che, per esempio, va da –10 a 10, dove il segno negativo indica una probabilità di correlazione inversa. L'uso dei fattori ponderali permette di ragionare con una logica "sfumata" e di risolvere le contraddizioni fra evidenze contrastanti.

Il secondo livello di rappresentazione della conoscenza è costituito da insiemi di regole che correlano le proposizioni. Per citare un esempio molto semplice:

Se sintomo-di (*x*, *y*) e presenta(*z*, *x*) allora soffre-di(*z*, *y*)

ciò significa che se *x* è sintomo della malattia *y* e il paziente *z* presenta il sintomo *x* allora il paziente soffre della malattia *y*.
Questo è un esempio particolare della costruzione più generale:

Se *A* e *B* allora *C*

dove *A*, *B* e *C* sono proposizioni. Costruzioni di questo tipo possono essere manipolate da un insieme di regole generali di inferenza quali:

Se *A* implica *B*, e *B* è falso, allora *A* è falso.

Queste regole generali possono essere utilizzate per generare nuove proposizioni o regole partendo da un insieme già esistente di proposizioni e regole che le correlano, o per dimostrare la correttezza di proposizioni o regole alla luce di regole e proposizioni date. Le regole di inferenza e i procedimenti formali per applicarle sono noti come *calcolo dei predicati*.
In qualsiasi situazione, il problema è sapere in quale successione applicare le regole di inferenza a un determinato insieme di proposizioni e di regole, senza dar luogo a un'esplosione combinatoria. Ad esempio, se si sa che

A implica *B*

allora qualsiasi proposizione ulteriore può essere inclusa in costruzioni quali:

A e *C* implica *B*,
A e *D* implica *B*,

ecc. Questo determina il terzo livello di rappresentazione della conoscenza in un computer: una strategia per controllare l'applicazione delle regole di inferenza a particolari regole pertinenti alla situazione. Lo sviluppo di strategie efficaci è stato

uno dei maggiori problemi per i ricercatori che si occupano di intelligenza artificiale.
Ci sono stati due modi generali di affrontare questo problema. Il primo è stato quello di attaccarlo in termini generali, e di cercare di sviluppare strategie coerenti per trarre inferenze da insiemi di regole. Tutto questo si è svolto entro i limiti della logica formale e ha portato, fra l'altro, allo sviluppo del linguaggio di programmazione Prolog. L'altro tipo di impostazione, sviluppare tecniche specifiche di inferenza che fossero applicabili in alcuni campi, fa parte dell'attività svolta sull'intelligenza artificiale, e ha portato allo sviluppo dei sistemi esperti. Entrambe queste linee di sviluppo, che verranno riprese negli ultimi paragrafi di questo libro, sono essenziali alla quinta generazione di computer. Attualmente ci sono parecchi sistemi di logica ognuno con il proprio insieme di assiomi e teoremi, e ognuno applicabile a una classe di problemi dell'intelligenza artificiale (Turner, 1984).
Edward De Bono (De Bono, 1969 e 1985) ha messo in evidenza una grave difficoltà insita in qualsiasi rappresentazione della conoscenza per il calcolatore. Tutte le tecniche di rappresentazione della conoscenza attualmente in via di sviluppo cercano di adattare la conoscenza a strutture fisse all'interno di un computer. Questa rappresentazione statica è in contrasto con i processi dinamici dell'intelligenza umana, dove le forme e le associazioni della conoscenza mutano costantemente, soprattutto durante il processo di apprendimento. Secondo le parole di De Bono: "Un vero computer pensante dovrà essere un sistema a autorganizzazione, il che è ben diverso dal concetto che normalmente abbiamo di sistemi informativi".

2.4 Programmi di gioco

Quello dei giochi è un settore in cui i computer hanno raggiunto livelli che mai si sarebbero potuti prevedere. Vent'anni fa, infatti, ben poche persone avrebbero preso seriamente in considerazione l'idea di poter giocare a scacchi con un computer; oggi, invece, i grossi computer stanno raggiungendo livelli da gran maestri, mentre i microcomputer sono più che all'altezza di disputare una partita a un discreto livello. Nel momento in cui scrivo, il campione del mondo di backgammon è un sistema computerizzato. Il successo riscosso dal software per i giochi è dovuto alla capacità di comprendere chiaramente alcune delle strategie comunemente usate nei giochi, alla grande capacità di memoria e alle veloci unità di elabo-

razione degli elaboratori elettronici moderni — in altre parole, alla combinazione della forza bruta con l'intelligenza.
Quale esempio di metodo basato sulla forza bruta, il gioco "punto e croce" prevede soltanto qualche centinaio di configurazioni possibili. Un computer alle prese con questo gioco può rapidamente considerare tutte le conseguenze di ciascuna mossa prima di effettuare la mossa successiva, fino alla fine del gioco, e quindi scegliere la mossa più favorevole. Nel gioco degli scacchi questo non è possibile neppure con i più potenti elaboratori esistenti, dato che il numero di mosse possibili si moltiplica troppo rapidamente. Il miglior programma per giocare a scacchi riesce a ottenere il giusto equilibrio tra l'ampiezza della ricerca (il numero delle possibili mosse studiate) la profondità della ricerca (il numero delle mosse conseguenti studiate per ogni possibilità) e misure applicate nella valutazione delle mosse.
Un esempio di valida strategia utilizzata nei programmi gioco è la regola minimax. Ogni mossa possibile viene valutata in base ai potenziali vantaggi che porterebbe all'altro concorrente, dando per scontato che l'avversario risponderebbe con la mossa per lui più vantaggiosa. La mossa che minimizzi il beneficio dell'avversario, partendo dal presupposto che egli sappia trarre il maggior vantaggio possibile da tale mossa, è quella scelta. Da qui il nome di minimax. Per quanto la regola minimax, in teoria, richieda che si esaminino fino alla fine del gioco le conseguenza di ciascuna mossa, sono state messe a punto una serie di tecniche per "potare" l'albero di ricerca dopo aver stabilito che un certa mossa è inferiore a un'alternativa già presa in considerazione (Boden, 1977). Viene stabilito un limite arbitrario alla "profondità" della ricerca, in funzione della potenza del computer che sta giocando. In molti giochi è stato dimostrato che la strategia minimax è troppo prudente, e si usano strategie più rischiose.

2.5 Programmi di ragionamento

I programmi di gioco non sono che un caso particolare della più generica classe dei programmi di ragionamento. Questi sono stati utilizzati per risolvere sia problemi quali il riconoscimento delle forme, ricorrenti nei test per valutare il quoziente di intelligenza, che problemi di logica formale. Le capacità e i limiti dei programmi di questo tipo sono state dimostrate chiaramente attraverso l'utilizzazione dei computer per dimostrare il famoso Teorema dei quattro colori (Appel e Haken,

1976). È risaputo da secoli che, per colorare una carta geografica in modo che due zone adiacenti non abbiano mai lo stesso colore sono sufficienti quattro colori diversi. Questo teorema è stato dimostrato con l'aiuto di un programma computerizzato nel 1976.
La dimostrazione consta di due parti. In primo luogo il programma genera circa 1800 elementi base della carta geografica e dimostra che tutte le carte geografiche non sono che il risultato di combinazioni topologiche di questi elementi. Quindi il programma prende in considerazione ogni elemento singolarmente e mostra che, in ogni caso, sono necessari non più di quattro colori perché ogni zona non abbia lo stesso colore di quelle a essa adiacenti. Per quanto molti matematici abbiano trovato da ridire sull'ineleganza della dimostrazione — richiede, infatti, molte sottodimostrazioni separate — e altri non la vogliono considerare, non si è mai trovato un metodo migliore e pertanto questa tecnica basata sul calcolatore rimane la dimostrazione accettata. Questa metodologia dimostrativa, in cui un problema viene ridotto a tanti piccoli sottoproblemi, può essere applicata anche ad altre situazioni, soprattutto quando si voglia dimostrare la correttezza o meno di un programma di calcolatore.

2.6 Riconoscimento del linguaggio naturale

Una delle conclusioni a cui siamo arrivati nel primo capitolo è che gli elaboratori non possono interpretare, in termini generali, i continui mutamenti del linguaggio naturale. Ciò nonostante, essi possono gestire singole parole e frasi e, in alcuni settori specifici, passi più lunghi in linguaggio naturale. Uno degli obiettivi fondamentali dello sviluppo della quinta generazione di computer è riuscire a migliorare la capacità di risposta degli elaboratori elettronici ai linguaggi naturali, al punto di rendere possibili molte applicazioni a controllo vocale.
Gran parte dell'attività svolta sul riconoscimento del linguaggio naturale da parte del computer trova origine nell'opera del linguista Noam Chomsky (ad esempio, Chomsky 1965 e 1968). Chomsky ha dimostrato che i linguaggi possono essere classificati in diversi livelli, a seconda della loro complessità latente. Al livello più alto (le grammatiche sensibili al contesto) si trovano i linguaggi naturali mentre al livello immediatamente inferiore (le grammatiche libere dal contesto) vi sono i linguaggi di programmazione. Chomsky, inoltre, ha mostrato che esiste una

"struttura profonda" sottostante alle frasi dei linguaggi naturali e che frasi equivalenti nei diversi linguaggi naturali hanno la stessa struttura profonda.
Quando soltanto si pensa alle difficoltà solitamente incontrate nel trattare con le lingue naturali, si può ben capire quanti grossi problemi sorgano nel tentare di trasferire una certa capacità linguistica a una macchina. Il primo è un problema di sintassi. Le lingue naturali sono fatte di strutture, quali le frasi, che vengono costruite secondo precise regole formali. Ad esempio, la frase:

Il gatto è seduto sullo zerbino

può essere scomposta (o analizzata) in:

<soggetto> <verbo> <oggetto>

dove <soggetto> ("il gatto") può essere ulteriormente analizzato come

<articolo> <nome>

ecc. Il problema di questo tipo di analisi è che le regole per la costruzione delle frasi sono molto complesse, ci sono molte eccezioni, e le regole vengono gradualmente modificate con l'evolvere della lingua.
Comunque, per capire un brano, l'analisi sintattica non è sufficiente. La semantica, cioè il significato di un elemento, dipende tanto dal contesto e da ciò che è stato detto prima, quanto dal significato delle singole parole. Molto spesso una diversa interpretazione di una singola parola può alterare il significato dell'intero brano. Prendiamo, per esempio, una frsase come

La vecchia porta la sbarra

che è un vero capolavoro di ambiguità. Un essere umano in genere non ha molta difficoltà a vedere le sue possibili interpretazioni, che danno anche due strutture sintattiche diverse. Comunque, senza informazioni contestuali, un uomo, e tanto meno un computer, non riuscirebbe mai a decidere correttamente quale tra le due possibili interpretazioni debba essere applicata, caso per caso.
Un esempio dei progressi compiuti nel riconoscimento della voce umana è il

sistema sviluppato da Terry Winograd, chiamato Shrdlu (Winograd, 1972). Shrdlu ha un ambiente composto da una superficie piana (il "tavolo"), una scatola e alcuni blocchi di diverse forme (cuboidi o piramidali), dimensioni e colori. C'è il braccio di un robot immaginario, in grado di raccogliere un blocco e di spostarlo in un'altra posizione. L'interazione con l'utente avviene per mezzo di uno schermo grafico che permette di visualizzare la posizione dei blocchi, e di messaggi scambiati in inglese con il braccio del robot. Il robot può agire in base alle istruzioni e rispondere alle domande sullo stato dei blocchi o sulle sue azioni passate. Dato che il sistema possiede una grande quantità di informazioni contestuali sullo stato dei blocchi e sulle regole che determinano il suo comportamento (ad esempio le piramidi possono essere appoggiate su blocchi, ma non viceversa), e registra tutte le azioni precedentemente intraprese con l'utente, esso è in grado di interpretare istruzioni che potrebbero essere molto ambigue. Ad esempio:

Utente: Solleva un blocco rosso.
Shrdlu: OK. (Lo schermo grafico mostra che il blocco rosso viene sollevato.)
Utente: Metti l'oggetto dentro la scatola.
Shrdlu: Per "oggetto" suppongo che tu intenda il blocco che sto tenendo. OK. (Lo schermo grafico mostra che il blocco viene messo nella scatola. Se nella scatola fosse già contenuta una piramide, questa sarebbe stata precedentemente spostata.)
Utente: Solleva la piramide.
Shrdlu: Non capisco quale piramide intendi. (Non viene intrapresa alcuna azione, dato che l'utente non ha precedentemente menzionato alcuna piramide.)

Il sistema di Winograd comprende tre programmi che interagiscono in modo piuttosto complesso. Il primo è un sistema di analisi sintattica che interpreta e costruisce la sintassi degli enunciati scambiati con l'utente. Il secondo è un sistema semantico che deduce il significato degli scambi verbali. Il terzo è un sistema deduttivo che applica le istruzioni ai blocchi e determina le risposte del braccio del robot. Per capire il significato di ogni enunciato, il sistema semantico richiede informazioni sia dal sistema sintattico che dal sistema deduttivo.
Shrdlu illustra sia il punto forte sia il punto debole nell'interpretazione del linguaggio naturale da parte di un elaboratore elettronico. Da un lato, esso può rispondere a una gamma molto ampia di richieste e può destreggiarsi con parole quali "questo",

"quello" e "esso". Dall'altra, è limitato al suo mondo circoscritto dei blocchi colorati. Per gestire questo suo piccolo mondo è necessario un programma molto grosso e complesso. Esso, comunque, indica il tipo di approccio — analisi sintattica e semantica con riferimenti a una sottostante base di conoscenza— necessario per mettere a punto dei sistemi di elaborazione del linguaggio naturale.

2.7 Rionoscimento dell'immagine

Nei neonati è necessario correggere quanto prima eventuali difetti visivi altrimenti, anche se venissero curati in un secondo tempo, il loro cervello non imparerebbe mai a interpretare i segnali visivi. Questo indica chiaramente che la vista riguarda solo in minima parte gli occhi, e in massima parte il modo in cui il cervello elabora l'informazione visiva che lo raggiunge. Non è infatti un caso che la parola "vedere" venga spesso utilizzata con il significato di "capire".
I computer possono produrre immagini grafiche di sorprendente complessità. Non sono, però, altrettanto sorprendenti i progressi compiuti per interpretare la forma visiva fornita, ad esempio, da una cinepresa. Le regole base dell'interpretazione visiva sono la geometria euclidea e la topologia (la teoria delle superfici connesse ecc.), ma queste non sono un'utile base teorica per la visione da parte del computer. La geometria euclidea indica in modo preciso come costruire una rappresentazione bidimensionale di una scena tridimensionale. Ma per il computer, il problema della vista è essenzialmente l'opposto — come ricreare una scena tridimensionale e riconoscere gli oggetti in essa contenuti partendo da una rappresentazione bidimensionale. Come afferma Edward De Bono: "Il riconoscimento delle forme nel cervello è estremamente semplice: per il modo stesso in cui è fatto, il cervello non può fare a meno di riconoscere le forme. Al contrario, il riconoscimento delle forme da parte dei computer è estremamente difficile, dato che l'approccio è opposto. L'architettura del cervello è molto diversa dall'architettura dei computer" (De Bono, 1985).
L'intelligenza artificiale ha affrontato questo problema costruendo degli insiemi di "regole empiriche" in base a cui interpretare particolari forme visive, e applicandoli alle varie situazioni specifiche. La maggior parte dei sistemi comincia con l'identificazione dei bordi dei vari oggetti (compito molto più difficile di quanto possa sembrare), quindi ne deduce le forme e le altre proprietà. Il più delle volte, i

programmi sviluppati permettono di decidere correttamente se la silhouette di una persona corrisponde a quella di un uomo o di una donna, e identificano il tipo di blocchi utilizzati nel sistema Shrdlu descritto sopra. Alla Brunel University è stato messo a punto un sistema hardware specializzato, noto col nome di Wisard, in grado di riconoscere l'identità di un individuo trovando la corrispondenza tra l'immagine della faccia di una persona presente sul video e quella memorizzata assieme a molte altre. Sta ora diventando chiaro che la chiave all'elaborazione intelligente dell'immagine non è che un'elaborazione in parallelo a un livello tanto alto al punto, forse, di avere un microprocessore per ogni pixel dello schermo grafico (Duff, 1982).

2.8 Sistemi esperti

Nell'ambito dell'intelligenza artificiale, il settore in cui si sono raggiunti i migliori risultati è, senza dubbio, quello dei sistemi esperti. Un sistema esperto è un programma che, in un determinato campo, raggiunge un livello di competenza pari a quello di un essere umano esperto. I sistemi esperti sono prodotto di quella linea di ricerca che ha trascurato la ricerca di soluzioni generali ai problemi dell'intelligenza, e si è dedicata piuttosto al trasferimento di aspetti specifici dell'intelligenza umana ai computer. I sistemi esperti sono una delle aree principali di applicazione dei computer della quinta generazione, e verranno trattati ulteriormente nel paragrafo 10.6.

2.9 Linguaggi di programmazione per l'intelligenza artificiale

Come abbiamo già visto alla fine del primo capitolo, ci sono state due grandi linee di impostazione dello sviluppo del concetto di computer. Una è quella della macchina astratta, l'altra quella della programmazione funzionale. Quest'ultima è diventata parte integrante nello sviluppo dei linguaggi di programmazione negli studi sull'intelligenza artificiale. È probabile che molti di questi linguaggi influenzeranno notevolmente i linguaggi di programmazione dei computer della quinta generazione. Essi verranno trattati nel capitolo 7.

2.10 Conclusione

Nonostante le difficoltà dei concetti fondamentali e le discordanze tra gli esperti del settore, notevoli sono i progressi compiuti nel campo piuttosto astruso e a volte controverso dell'intelligenza artificiale. In particolare, è stata dimostrata, in alcuni campi particolari, la possibilità di trasferire un certo grado di intelligenza dalla persona al computer. Secondo molti tra i maggiori esperti del settore, una delle sfide principali è riuscire ad automatizzare il buon senso umano. Con l'aumentare dell'importanza attribuita all'intelligenza artificiale quale fondamento delle basi teoriche della quinta generazione di computer, è probabile che l'andamento dei progressi sull'intelligenza artificiale acceleri notevolmente nei prossimi anni.

3

I programmi della quinta generazione

L'idea di un computer della quinta generazione venne ampiamente divulgata per la prima volta a un congresso internazionale a Tokyo, nell'ottobre 1981 (Moto-Oka, 1982). Sia la denominazione "computer della quinta generazione" quanto i progetti rivoluzionari che essa comportava erano di origine giapponese — prodotto di due anni di ricerche di alto livello presso il Japan Information Processing Development Centre (Centro di sviluppo dell'elaborazione dell'informazione giapponese, JIPDEC). La reazione dei presenti — figure di primo piano del mondo industriale informatico provenienti da gran parte dei paesi occidentali — fu simile a quella dal lancio dello Sputnik sovietico nel 1957. Fu subito chiaro a tutti che i computer della quinta generazione rappresentano una rottura fondamentale con l'elaborazione dati tradizionale e che la realizzazione, anche parziale, di quei progetti ambiziosi renderà obsoleta la maggior parte degli elaboratori elettronici tradizionali. Il significato industriale, economico e politico di queste possibilità non sfuggì a nessuno dei presenti né alle organizzazioni commerciali e ai paesi che essi rappresentavano.

Durante il congresso venne lanciata l'idea di lavorare in collaborazione internazionale per far sì che il concetto ancora teorico di computer della quinta generazione potesse essere realizzato in prodotti commerciabili. In seguito, però, questa idea venne abbandonata, a favore di un'attività di ricerca e di sviluppo a livello regionale o nazionale. Entro il 1984 erano già stati avviati cinque programmi principali: il

programma giapponese Icot, i progetti MCC e Darpa negli Stati Uniti, l'iniziativa Esprit all'interno della Comunità Economica Europea e il programma Alvey in Gran Bretagna. I rapporti tra i vari gruppi sono sempre stati un misto di collaborazione e di competizione. C'è infatti un libero flusso di informazioni teoriche attraverso pubblicazioni e convegni internazionali (ad esempio: SPL International, 1982 e 1983, e Scarrott, 1983), ma la maggior parte dei progetti sono tenuti a rispettare notevoli restrizioni circa la diffusione di dettagli specifici sui prodotti in via di sviluppo.

3.1 Giappone: il programma Icot

> Se il progetto avrà successo, sarà come se avessimo preso tre piccioni con una fava. In primo luogo, esso potrà soddisfare la necessità di un nuovo tipo di computer che si determinerà in seguito all'utilizzo sempre più sofisticato delle informazioni. In secondo luogo, contribuirà alla crescita dell'industria elettronica che, in futuro, diventerà il nucleo della struttura industriale giapponese. Infine, contribuirà notevolmente all'internazionalizzazione del Giappone (Osamu Seki, Direttore, Electronics Policy Division, MITI, citato da Kikuchi, 1983).

Tra tutti i progetti di sviluppo della quinta generazione, quello giapponese è il più ambizioso e quello maggiormente accentrato. L'iniziativa giapponese è coordinata dall'Institute for New Generation Computer Technology (Istituto di tecnologia per la nuova generazione di computer, ICOT), fondato nell'aprile 1982 sotto la direzione di Kazuhiro Fuchi, con un budget di 855 milioni di dollari per un programma decennale in tre fasi (Fuchi, 1983). Il progetto è considerato un elemento decisivo per l'andamento dell'industria giapponese e per la continua prosperità del paese negli anni novanta e anche oltre. Sono state identificate sette ampie aree di ricerca (figura 3.1) che rappresentano i principali requisiti di progettazione dei computer della quinta generazione. Ciascuna di esse è suddivisa in più specifici campi di ricerca e in alcuni stadi prevede la possibilità di trasferire i risultati intermedi. Una delle prime cose da realizzare è incorporare i nuovi sviluppi del software nella struttura hardware: un aspetto importante del progetto è il sistema di progettazione assistita dall'elaboratore (CAD) per chip ad altissima integrazione (VLSI).

Sistemi applicativi di base
- 1.1 Sistema di traduzione computerizzata
- 1.2 Sistema di risposta alle domande
- 1.3 Sistema applicato di comprensione del parlato
- 1.4 Sistema applicato di comprensione dell'immagine e del disegno
- 1.5 Sistema applicato di risoluzione di problemi

Sistemi di software di base
- 2.1 Sistema di organizzazione della base di conoscenza
- 2.2 Sistema di risoluzione dei problemi e delle inferenza
- 2.3 Sistema di interfacce intelligenti

Nuove architetture avanzate
- 3.1 Elaboratore di programmazione logica
- 3.2 Elaboratore funzionale
- 3.3 Elaboratore dell'algebra relazionale
- 3.4 Elaboratore di supporto dei tipi di dati astratti
- 3.5 Elaboratore a flusso di dati
- 3.6 Nuovo elaboratore di Von Neumann

Architettura di funzioni distribuite
- 4.1 Architettura di funzioni distribuite
- 4.2 Architetture di reti
- 4.3 Elaboratore a base di dati
- 4.4 Elaboratore per il calcolo numerico ad altissima velocità
- 4.5 Sistema di comunicazione uomo-macchina ad alto livello

Tecnologia ad altissima integrazione (VLSI)
- 5.1 Architetture VLSI
- 5.2 Sistemi CAD intelligenti ad altissima integrazione

Tecnologia
- 6.1 Sistema di programmazione intelligente
- 6.2 Sistema di progettazione della base conoscitiva
- 6.3 Tecnologia di sistematizzazione per architetture di computer
- 6.4 Sistema di database e di database distribuiti

Tecnologia di supporto allo sviluppo
- 7.1 Sistema di supporto allo sviluppo

Figura 3.1 Il programma Icot per lo sviluppo dei computer della quinta generazione.

La prima delle tre fasi è costituita da studi di fattibilità e dalla progettazione degli strumenti da utilizzare nelle ricerche future. Per intraprendere gli studi preliminari, a ogni ricercatore è stato fornito un computer da tavolo a inferenza sequenziale (PSI). In un primo tempo, per le ricerche sul software si è preferito il linguaggio di programmazione Prolog, mentre per i modelli dei componenti hardware ci si è orientati verso le architetture a flusso di dati. Nella seconda fase, iniziata nel 1985, si mettono alla prova i due meccanismi base dei computer della quinta generazione: le basi di conoscenza e motori inferenziali. Nella fase finale, che inizierà nel 1989, si costruirà il primo prototipo di un computer della quinta generazione.
Come è consuetudine in Giappone, c'è una stretta collaborazione tra i ricercatori del centro ICOT, molti dei quali dipendono dalle società di tecnologia dell'informazione (IT) e l'industria informatica e delle telecomunicazioni. Le ricerche di base e lo sviluppo del primo livello si svolgono all'interno dell'ICOT, mentre lo sviluppo di un prodotto specifico è appaltato alle società di informatica e di elettronica (Giapponesi). Il programma ICOT viene amministrato dal Ministero del Commercio Internazionale e dell'Industria (MITI), lo stesso gruppo di coordinamento alla base della rapida crescita economica del Giappone nel dopoguerra.
Nei paesi occidentali, il progetto ICOT è considerato un importantissimo passo avanti nel passaggio dal tradizionale ruolo del Giappone quale perfezionatore e produttore di massa di una tecnologia già affermata. Sono emersi alcuni dubbi circa la capacità dei ricercatori, abituati a lavorare per raggiungere mete già stabilite e in tempi brevi, di raggiungere il livello di innovazione richiesto. Anche nei paesi occidentali vi è un certo scetticismo sulle prestazioni specifiche dei vari elementi di un computer della quinta generazione che i gruppi di ricerca giapponesi intendono realizzare. Comunque, se il programma ICOT dovesse venir realizzato anche solo parzialmente, il Giappone occuperebbe certamente il primo posto nella classifica degli innovatori della tecnologia dell'informazione per molti anni a venire.

3.2 Gli Stati Uniti: Darpa e il MCC

Negli Stati Uniti la risposta all'iniziativa giapponese della quinta generazione di computer è stata ostacolata dalla presenza di vincoli tradizionali e legali sulle pratiche commerciali. L'idea di una società in collaborazione nazionale nella ricerca avanzata ha solo un precedente: il programma Nasa per mandare il primo

uomo sulla luna. Uno scontro tra le società rivali nello sviluppo avanzato dei computer potrebbe violare le leggi antitrust che costituiscono la base del capitalismo americano. Fanno eccezione tutte le attività con possibili applicazioni nel settore della difesa. È proprio per questo che, negli Stati Uniti, gran parte dell'attività di ricerca e sviluppo della quinta generazione viene coordinata dalla Defence Advanced Research Project Agency (Agenzia per il Progetto di Ricerche Avanzate per la Difesa, Darpa), con un bilancio di 50 milioni di dollari nel 1984, e una possibilità di ulteriori finanziamenti fino a un miliardo di dollari. Darpa è un'agenzia di investimenti e specificazione che convoglia le richieste dell'istituto della difesa nei vari contratti con le società commerciali per realizzarle. Fra i progetti che rispecchiano il concetto di un computer della quinta generazione sponsorizzati da Darpa vi sono il riconoscimetno della voce e la sintesi del parlato, l'analisi del linguaggio naturale, le basi di dati relazionali, l'elaborazione delle immagini e la progettazione di semiconduttori avanzati.

L'altra iniziativa statunitense della quinta generazione è un'impresa commerciale, la Microelectronic and Computer Technology Corporation (Società di microelettronica e tecnologia dei calcolatori, MCC), fondata da William Norris della Control Data Corporation e con un budget superiore ai 50 milioni di dollari (Gannon, 1983). La MCC è un consorzio di società di tecnologia dell'informazione, ognuna delle quali paga una quota d'entrata pari a un minimo di 100.000 dollari per poter accedere ai risultati delle ricerche e ai progetti dei prototipi. Tra i soci figurano la Digital Equipment Corporation, la National Cash Register e i produttori di chip Motorola e National Semiconductor. L'obiettivo della MCC è svolgere ricerche applicate e sviluppare prodotti competitivi su vari aspetti dei computer della quinta generazione, con particolare interesse per le tecniche di progettazione assistita dall'elaboratore per la produzione di chip ad altissima integrazione, l'elaborazione delle immagini e i sistemi esperti. Direttore del progetto è Robert Inman, che in precedenza ha prestato servizio presso la marina americana e la CIA. L'IBM non fa parte della MCC, ma certamente ha destinato parte dei 2 miliardi investiti annualmente in ricerche e sviluppo ad attività sulla quinta generazione.

Un settore di sviluppo dei computer della quinta generazione in cui gli Stati Uniti sono decisamente all'avanguardia è costituito dalle ricerche di base sull'intelligenza artificiale. Negli anni settanta in Gran Bretagna, in seguito a una diminuzione dell'interesse e dei finanziamenti verso questa attività, molte celebrità del mondo accademico si sono trasferite negli Stati Uniti, rinforzando così le già

efficienti squadre di ricerca in molte delle principali università americane. Inoltre, grazie ai forti legami tradizionali tra il mondo accademico e quello industriale americano, le nuove scoperte sono presto a disposizione degli altri soci che aderiscono all'iniziativa di sviluppo dei sistemi della quinta generazione.

3.3 La CEE: Esprit

Nel 1982 all'incontro dei vertici a Versailles, i leader della Comunità Economica Europea, su suggerimento del Presidente Mitterand, hanno concordato di esaminare la possibilità di mettere a punto un programma in collaborazione per svolgere attività di ricerca e sviluppo nei settori avanzati della tecnologia informatica. Nacque così l'European Strategic Research Programme in Information Technology (Programma europeo di ricerca strategica nella tecnologia dell'informazione, Esprit), progettato da un gruppo informale di importanti società europee e dal funzionario per la tecnologia dell'informazione della CEE, il visconte Etienne Davignon. Ci volle più di un anno perché il suo budget di 450 milioni di lire venisse approvato da tutti gli strati della burocrazia della CEE; alla fine, comunque, furono avviati un certo numero di progetti. Rispetto all'iniziativa giapponese, il programma decennale prevede di coprire una gamma più vasta di progetti. Infatti, oltre agli aspetti dell'intelligenza artificiale, sono previste altre applicazioni, quali l'office automation. È probabile che il programma Esprit confluisca nella più ampia iniziativa Eureka, il cui scopo è quello di coordinare ricerca e sviluppo in molti settori di alta tecnologia.

Il programma Esprit ha un piccolo esecutivo che discute con la Comunità eventuali proposte di ricerche e sviluppo avanzate dalle aziende e dagli istituti accademici o, preferibilmente, da consorzi internazionali di tali organizzazioni. I progetti approvati vengono finanziati fino al 50 per cento dalla CEE a condizione che i risultati ottenuti vengano sfruttati all'interno della Comunità. Caratteristico dei progetti è un piano di studio della struttura dell'ambiente di sviluppo del software per i programmi di logica di cui ci si occupa presso l'Imperial College di Londra. Gran parte dei fondi rimanenti sono stati concessi al produttore di semiconduttori inglese, Plessey, per attività di ricerca sui chip ad altissima integrazione e sulla progettazione e sulla fabbricazione assistita dall'elaboratore (CAM/CAD).

Il progetto Esprit è stato accusato di eccessiva ingerenza politica e burocratica e di

discriminazione dei confronti delle piccole aziende nella sua politica di concessione di sovvenzionamenti. C'è inoltre un'incertezza endemica riguardo a questi finanziamenti, dato lo stato di precarietà in cui si trovano le finanze della CEE. Comunque, sono tutti d'accordo sul fatto che, per far sì che le società e istituzioni accademiche degli stati membri continuino a far parte della comunità informatica mondiale, sia necessaria un'iniziativa a livello europeo nel settore della quinta generazione di computer.

3.4 Regno Unito: il programma Alvey

> Il problema che abbiamo di fronte è chiaro. Possiamo cercare di essere all'avanguardia in queste tecnologie, o possiamo decidere di contare sulla tecnologia importata; oppure possiamo scegliere di non partecipare a questa iniziativa. Non riteniamo che quest'ultima sia una scelta possibile. (Rapporto del Comitato Alvey, 1982).

La risposta del Regno Unito all'iniziativa giapponese della quinta generazione fu nominare una commissione che svolgesse ricerche sul problema e pubblicasse poi un rapporto. Presieduta da John Alvey, direttore della British Telecom Technology, questa commissione comprendeva importanti industriali e personaggi eminenti del mondo accademico ed esaminava gli eventuali risultati conseguiti nei vari settori dal Regno Unito. Il rapporto, pubblicato nell'ottobre 1982, proponeva un programma nazionale quinquennale nel settore dell'informatica avanzata, con un budget di 350 milioni di sterline inglesi. Esso invitava a compiere "uno sforzo in collaborazione decentralizzata tra l'industria, il settore accademico e altre organizzazioni di ricerca, col supporto del Governo" (Alvey Report, paragrafo 8.1). Nell'aprile del 1983 il progetto venne approvato dal Governo britannico quasi senza riserve e per la sua realizzazione venne stanziato un fondo statale di 200 milioni di sterline inglesi; il bilancio sarebbe stato presentato dagli industriali che aderivano al programma.
Venne nominato un piccolo esecutivo sotto la direzione di Brian Oakley, assistito da sette direttori di settore dipendenti di industrie informatiche. Il loro compito era stabilire obiettivi generali per le aree di sviluppo coperte dal programma, e assegnare fondi statali per un ammontare massimo corrispondente al 50 per cento del

costo del progetto approvato. Il primo progetto comprendeva sistemi di dimostrazione per sottoporre a test le idee della quinta generazione per verificare la possibilità di una loro concreta realizzazione.
L'Alvey Report identifica quattro fondamentali tecnologie necessarie alla realizzazione dei computer della quinta generazione: ingegneria del software, chip ad altissima integrazione, interfacce utente intelligenti e sistemi intelligenti basati sulla conoscenza (IKBS). Queste quattro linee di sviluppo sono rispecchiate nella struttura generica dell'Alvey Directorate. I fondi Alvey sono stati assegnati esclusivamente a progetti intrapresi da parte di società o consorzi inglesi, e tutti i risultati ottenuti devono essere realizzati in prodotti all'interno della Gran Bretagna. C'è una certa collaborazione tra il progetto Alvey e l'Esprit per far sì che non vengano raddoppiati gli sforzi tra i due programmi.
Nel Regno Unito sono in corso parecchi progetti di ricerca e sviluppo che, pur non essendo stati inizialmente sovvenzionati dal programma Alvey, rientrano nella quinta generazione di computer. Tra questi vanno inclusi il prototipo di un computer a flusso di dati, messo a punto all'Università di Manchester nel 1976, e un elaboratore, realizzato all'Imperial College, per la progettazione dei linguaggi di programmazione avanzati (paragrafo 5.6). (In un secondo tempo, a entrambi vennero concessi dei sovvenzionamenti Alvey.) Donald Michie, membro del gruppo Colossus e pioniere nel settore dell'intelligenza artificiale, ha fondato il Machine Intelligence Research Affiliates (Mira), un consorzio di sette società di tecnologia informatica comprendente anche la ICL e molte società petrolifere, per sviluppare prodotti di tecnologia dell'informazione che utilizzassero i concetti e le tecniche dell'intelligenza artificiale. A Londra, in collaborazione con l'Imperial College è sorto l'Imperial Software Technology (IST), per mettere a disposizione dell'industria l'esperienza raggiunta nell'ingegneria del software. Il Transputer Inmos (paragrafo 5.7) e i linguaggi di programmazione Occam a esso associati (paragrafo 7.4), sono progettati tenendo conto delle applicazioni della quinta generazione.
Il programma Alvey è stato criticato per non aver raggiunto il livello di alta perfezione del modello Icot, e per il basso tetto di sovvenzionamenti pubblici che esso prevedeva, che effettivamente escludeva le piccole società dalla possibilità di aderire all'iniziativa. È certo, comunque, che l'ampia consultazione intrapresa per la stesura del rapporto Alvey ha dato luogo a un programma adatto alla reale situazione presente in Gran Bretagna nel settore dell'informatica e ha dato all'industria quell'impulso e quello scopo di cui c'era estremo bisogno. Si è

sottolineata l'importanza del progetto e dei suoi potenziali benefici nei confronti della società, e sono stati ampiamente dibattuti i rischi di un eventuale abuso della potenza della tecnologia informatica avanzata. Si stanno vedendo i primi frutti della collaborazione tra le aziende e i ricercatori del mondo accademico. Tale collaborazione è un requisito essenziale al progresso in un programma tanto complesso quanto il progetto della quinta generazione. Guardato in restrospettiva, il progetto Alvey può essere considerato un tentativo di dare un certo impulso alla ricerca e allo sviluppo nel settore dell'informatica in Gran Bretagna, in modo da costituire una solida base per l'attività futura, da svolgere senza i sussidi statali. Ci si aspetta che non ci sarà soluzione di continuità nella collaborazione stabilita nel corso del programma Alvey, soprattutto tra le organizzazioni industriali e accademiche.

3.5 Conclusione

L'entusiasmo per la quinta generazione, che ha tanto allettato la fantasia della comunità informatica all'inizio degli anni ottanta, sta ora leggermente calando date le enormi difficoltà che i gruppi di ricerca dei paesi aderenti si trovano a dover affrontare. Altri paesi, quali il Canada, hanno lanciato altri progetti nella stessa direzione ma di minor portata e sembra che anche l'Unione Sovietica sia impegnata in un programma equivalente. L'importanza che rivestono i programmi varia, in certa misura, da paese a paese: il progetto giapponese è considerato fondamentale per assicurare continuità al benessere economico e sociale del paese. In Gran Bretagna l'attività viene svolta tra la crescente preoccupazione che la bilancia commerciale del paese nel settore dell'informatica stia continuando a peggiorare e che il paese possa perdere completamente il suo posto nella "serie A" della tecnologia delle informazioni. Considerazioni simili sono valide per tutto il resto dell'Europa Occidentale. I programmi inglesi ed europei sono soggetti al controllo politico e hanno sovvenzionamenti e durata di vita limitati. Negli Stati Uniti il programma della quinta generazione rientra negli sforzi che le società di informatica continuano a sostenere per mantenere le loro posizioni di preminenza. La MCC ha il vantaggio di essere un'istituzione permanente in grado di ricorrere ai sussidi statali ogniqualvolta questi siano disponibili, e di non essere, allo stesso tempo, sottoposta al controllo ufficiale.
Nessuno degli esperti nutre delle illusioni circa la difficoltà del compito, e la

possibilità che potrebbe anche terminare con un fallimento. Nonostante ciò, l'approccio scaglionato adottato in tutti i programmi indica che gli sviluppi intermedi, non appena disponibili, possono essere assorbiti nel filone principale dell'informatica. Se uno dei gruppi nazionali, col progredire dei progetti della quinta generazione, dovesse trovarsi ad avere un vantaggio significativo sugli altri, quel paese o quella regione avrà la possibilità di diventare la potenza predominante nella tecnologia dell'informazione per un decennio o forse più.

4

Computer della quinta generazione: struttura generale

Il simbolo dell'elaborazione elettronica contemporanea — il supercalcolatore — è composto da banchi di unità di elaborazione ad alta velocità operanti in parallelo su matrici di numeri o strettamente accoppiati in un *pipeline*, con ulteriori unità di elaborazione che trasmettono l'informazione da e verso la memoria secondaria. L'assemblaggio di tutti i componenti raggiunge un'altissima capacità di elaborazione dei dati e utilizza chip ad altissima integrazione densamente impaccati. La potenza di elaborazione di questi superelaboratori continua ad aumentare in modo esponenziale, il loro volume fisico e la potenza assorbita dominuiscono con ogni nuovo modello, e il loro prezzò si riduce in termini reali. Nonostante ciò, non assomigliano assolutamente ai computer della quinta generazione e non è neppure probabile che possano essere compatibili con essi. Inoltre, le loro ampie capacità di elaborazione dell'informazione appariranno del tutto insignificanti a confronto con le strutture presentate in questo capitolo, se queste saranno anche solo parzialmente realizzate.
A grandi linee, un calcolatore della quinta generazione è concepito come una "serie di macchine di database e di elaborazione parallela interconnesse, alle quali si accede per mezzo di un motore inferenziale intelligente che può (tra le altre cose), accettare problemi formulati per mezzo di enunciati in linguaggio naturale" (Bramer, 1984). La sua funzione principale non è l'elaborazione dell'informazione comunemente intesa, ma trarre inferenze da basi di conoscenza. Ciò significa

incorporare un grado molto più alto di intelligenza rispetto ai computer attuali, in alcune circostanze simile a quella di un esperto. Ci si aspetta, infatti, che l'applicazione principale dei computer della quinta generazione sia nella risoluzione di problemi altamente complessi che, quando affrontati da uomini, richiedono un alto grado di ragionamento, intelligenza ed esperienza. Per ottenere questo risultato sarà necessario incorporare nella struttura dei computer della quinta generazione tutto quello che già conosciamo sull'intelligenza artificiale — e molto altro ancora. La data fissata per la prima completa realizzazione di questi concetti è l'inizio degli anni novanta.

Nonostante le loro enormi capacità, i computer della quinta generazione sono fatti per essere usati da persone che non sono, necessariamente, specialisti del settore informatico. A tal fine, una delle maggiori linee di sviluppo della quinta generazione è costituita dalle interfacce utente intelligenti — interfacce che permettono all'utente di comunicare con la macchina in modo semplice e naturale. Questo, in qualche modo, è in contrasto con le attuali interfacce utente che si adattano più alle esigenze del computer che alle conoscenze, inclinazioni o abitudini di pensiero della persona che l'utilizza. Requisiti centrali delle interfacce utente intelligenti sono l'interazione per mezzo di un linguaggio naturale o di un'ampia parte di esso, e l'interfacciamento per mezzo di suoni e immagini.

I campi d'applicazione in cui si prevede di introdurre i computer della quinta generazione comprendono la medicina, la prospezione geologica, l'amministrazione sociale, l'organizzazione delle risorse umane e fisiche, gli studi umanistici e una serie di altre applicazioni che richiedono l'elaborazione del linguaggio e la traduzione. Caratteristiche comuni a tutti questi campi sono un grande patrimonio di conoscenze, in gran parte euristico o empirico, complessi principi di base, spesso compresi solo in parte, e tutti ricevono scarsi benefici dagli attuali sistemi di calcolo. Inoltre, tutti i campi d'applicazione rivestono un alto significato sociale, sia nelle nazioni industrializzate che nei paesi in via di sviluppo che, entro gli anni novanta, potrebbero avere un'infrastruttura tecnologica più solida di quella attuale e trovarsi così in posizione tale da poter trarre grossi benefici da questi sviluppi.

4.1 La struttura generale di un computer della quinta generazione

La struttura proposta per un computer della quinta generazione venne presentata

alla conferenza tenutasi a Tokyo nell'ottobre del 1981, che segnava l'inizio del programma giapponese. Essa è frutto di due anni di studi e ricerche presso il Japan Information Processing Development Centre (Centro di sviluppo dell'elaborazione delle informazioni giapponese, JIPDEC, 1981). Il suo quasi unico punto di somiglianza con i sistemi esistenti è l'essere basata su un'architettura ad altissima integrazione (VLSI). Per tutti gli altri aspetti, sia fisici che logici, rappresenta una svolta radicale rispetto all'elaborazione dati convenzionale. Per quanto molti esperti abbiano espresso riserve, il modello giapponese rimane un valido punto di partenza per la discussione e, se non in tutti i dettagli almeno nel suo concetto generale, è stato accettato dai gruppi di ricerca dei vari paesi, quale base per ulteriori sviluppi.
Nella figura 4.1 è rappresentata l'"immagine della configurazione base" di un computer della quinta generazione. In pratica, ogni implementazione verrà "strutturata in base alla sua funzione" secondo gli elementi di questo diagramma. È anche previsto che le macchine del tipo qui rappresentato saranno collegate in reti locali o geografiche, ognuna delle quali fornirà ulteriori livelli di elaborazione distribuita.
La figura 4.1 mostra il diagramma di un sistema, e cerca di evidenziare le relazioni operative tra un certo numero di elementi fisici e logici. Osservato "orizzontalmente", presenta un livello di hardware, uno di software e un'interfaccia esterna con i sistemi applicativi, come si poteva prevedere. Ma osservato "verticalmente", diventa chiaro che ogni aspetto della funzionalità di un computer della quinta generazione — inferenza e risoluzione di problemi, organizzazione della base di conoscenze e interfacce intelligenti — richiede propri meccanismi di supporto hardware e software. Questi meccanismi interagiscono a vari livelli, ma sono essenzialmente autonomi. L'estensione e la natura del parallelismo implicito in questo modello va ben oltre qualsiasi cosa attualmente utilizzata o in via di sviluppo riguardante gli elaboratori elettronici tradizionali.
Semplificando forse un po' troppo la situazione, i tre elementi "verticali" della figura 4.1 sono raffigurati nella figura 4.2. Questa illustra la struttura concettuale di un computer della quinta generazione, senza riportare i meccanismi di supporto che comporta. I tre aspetti del sistema — base di conoscenza, elaborazione inferenziale e interfacce intelligenti — verranno trattati più avanti.
L'operazione di elaborazione di base di un computer della quinta generazione non è il calcolo numerico, ma trarre inferenze logiche. Di conseguenza, i requisiti di prestazione sono espressi in inferenze logiche al secondo (lips: *logical inferences per second*).
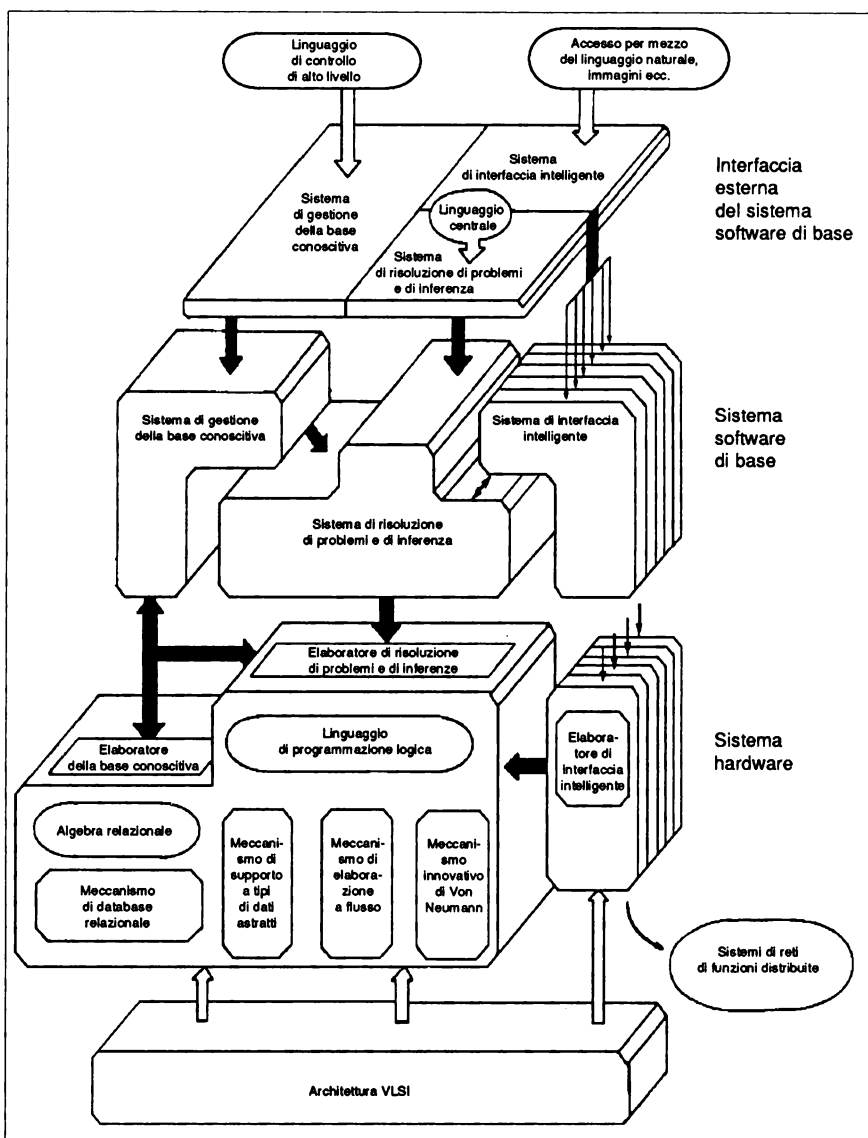
Fig. 4.1 - Immagine della configurazione base di un elaboratore della quinta generazione (Riprodotto su concessione del Japan Information Processing Development Center.)
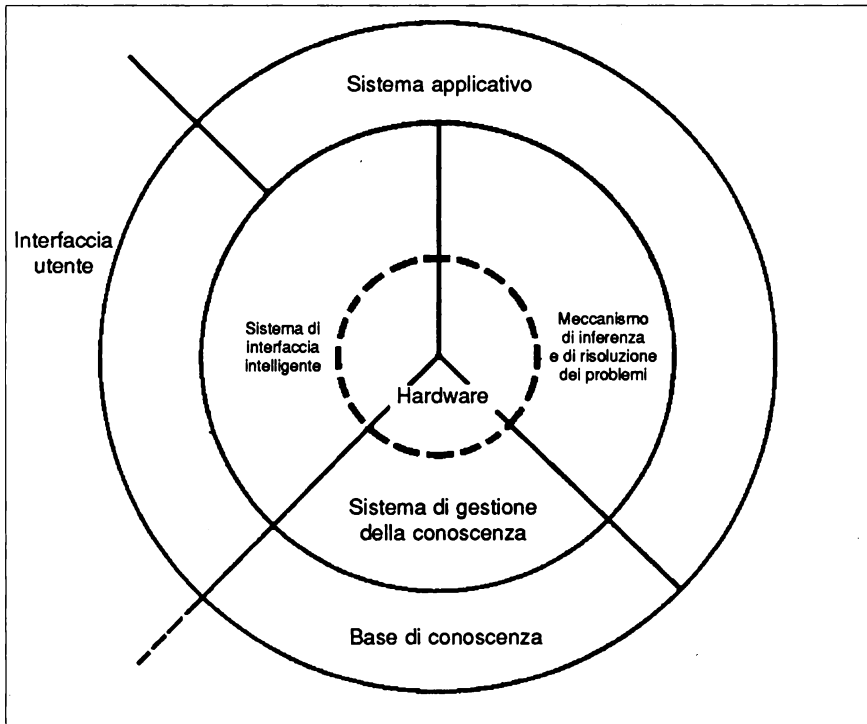
Fig. 4.2 - Gli elementi di un computer della quinta generazione.

L'obiettivo, per i meccanismi inferenziali e di risoluzione dei problemi di in computer della quinta generazion, sta tra 50 e 1000 milioni di lips. Per fare un confronto, i computer attuali operano tra 10 000 e i 100 000 lips. La capacità di memorizzazione delle unità di memoria che fanno da supporto alle varie unità di elaborazione saranno, probabilmente, tre o quattro ordini di grandezza maggiori delle memorie più grosse disponibili oggi. Per i sistemi di gestione delle basi di conoscenza l'obiettivo è una capacità pari a 1000 gigabyte (JIPDEC, 1981), contro i 100 megabyte di un attuale supercalcolatore.

4.2 Base di conoscenza

L'elemento di un computer della quinta generazione in un certo senso "più lontano" dall'utente è la base di conoscenza. Questa è concepita come un enorme bagaglio di conoscenze che può essere suddiviso in associazioni tra oggetti, regole per la formazione di nuove associazioni e strategie per l'applicazione di tali regole, come spiegato nel paragrafo 2.3. Caratteristica essenziale di una base di conoscenza è l'essere in grado di fare da supporto all'elaborazione in presenza di informazioni vaghe, incomplete o contraddittorie.
È probabile che una base di conoscenza sia un'estrapolazione di sviluppi attuali nelle basi di dati relazionali in cui i dati vengono memorizzati in tabelle, in cui ciascuna riga costituisce un esempio di una relazione tra le categorie dell'oggetto. Un esempio banale di questo tipo di base di dati è quello con due tabelle, una che associa l'indirizzo al nome, l'altra i numeri telefonici al nome:

Tabella 1

Nome	Indirizzo
A Rossi	Via Roma 12
M Bianchi	Via Trento 55
S Corsi	Via Grossi 12

Tabella 2

Nome	Numero telefonico
G Bacchi	06 568944
A Rossi	02 235689
S Corsi	049 623589

Un'operazione fondamentale delle basi di dati relazionali è la capacità di collegare le tabelle tramite di elementi comuni, generando così ulteriori associazioni. Nell'esempio precedente, le due tabelle potrebbero essere unificate in base al campo *Nome*, dando luogo così a una tabella combinata (contenente solo due righe) con *Nome*, *Indirizzo* e *Numero telefonico*. Si noti che l'operazione non viene applicata solo ai singoli elementi dei dati, ma a due intere tabelle, che, in pratica, possono contenere milioni di elementi; il risultato è la creazione di una terza tabella. La teoria alla base di questa forma di manipolazione dei dati è nota col nome di *algebra relazionale*. È probabile che i computer della quinta generazione richiedano meccanismi sia software che hardware per effettuare elaborazioni relazionali. Il software per l'elaborazione di basi di dati relazionali è semplice, ma la realizzazione dell'hardware per manipolare i dati non come elementi individuali, ma come tabelle complete in un unico passo procedurale, è oggetto di ricerca e sviluppo. Per

soddisfare i requisiti generali di prestazione dei computer della quinta generazione, sarà essenziale avere dell'hardware dedicato, rappresentato nella figura 4.1 come "meccanismo di base di dati relazionale".
Uno strumento concettuale di importanza vitale nella gestione di grandi quantità di dati complessi è la nozione di tipo di dati astratto. Ad esempio, le relazioni espresse nelle tabelle precedenti possono essere considerate in astratto, senza pensare alla loro realizzazione in una struttura di righe e colonne. Al livello superiore, un programma controlla l'elaborazione di questi oggetti astratti, mentre la realizzazione specifica e la registrazione di quegli effetti nella memoria centrale o secondaria vengono lasciati ai livelli inferiori dell'hardware e del software. La nozione di tipo di dati astratto permette di definire le proprietà logiche intrinseche degli oggetti dei dati, e di manipolare gli oggetti in base a queste proprietà. Si determina così una notevole semplificazione dei programmi, mantenendo separati gli aspetti fisici e logici di un sistema di programmazione. Applicata alle basi di dati, la nozione di tipi di dati astratti porta al concetto di indipendenza dei dati, secondo cui la struttura logica di una base di dati è completamente indipendente dalla sua rappresentazione su qualsiasi particolare supporto fisico.
Nei calcolatori di oggi i diversi livelli di astrazione dei dati vengono gestiti soprattutto da strati di software. Questo, però, non sarà più sufficiente per la quinta generazione. A un livello piuttosto basso, per far corrispondere le proprietà logiche dei tipi di dati astratti alle loro rappresentazioni fisiche, è infatti necessario un meccanismo di trasformazione implementato nella struttura hardware, illustrato nella figura 4.1 come un "meccanismo di supporto dei tipi di dati astratti", e costituisce un elemento chiave sia nell'elaborazione della base di conoscenza, che nell'elaborazione inferenziale.

4.3 L'elaborazione inferenziale

L'elemento "centrale" di un computer della quinta generazione è un meccanismo di elaborazione inferenziale, che interagisce con il sistema di interfaccia intelligente per comunicare con l'utente, e con il meccanismo della base di conoscenza per attingere alle risorse della conoscenza. Allo stesso tempo si dedica al suo compito fondamentale, la risoluzione di problemi che rientreranno nell'orbita dei computer della quinta generazione. Come per l'elaborazione della base di conoscenze, sono

necessari hardware e software dedicati.
Come già spiegato nel paragrafo 2.3, il compito fondamentale dell'elaborazione inferenziale è trarre conclusioni da situazioni reali. Queste possono essere "precise", come nell'esempio del paragrafo citato:

Se sintomo-di(*x*, *y*) e presenta(*z*, *x*) allora soffre di (*z*, *y*)

o possono includere un elemento di incertezza:

Se sintomo-di(*x*, *y*, probabilità>0.7) e presenta (*z*, *x*, probabilità>0.9) allora soffre-di (*z*, *y*, probabilità = 0.6)

Il software per l'elaborazione inferenziale sarà, probabilmente, basato su sviluppi nel campo della programmazione logica, oggi esemplificata dal linguaggio di programmazione Prolog. Un programma in Prolog contiene dichiarazioni delle relazioni esistenti tra i vari elementi e regole che permettono di trarre conclusioni dalle relazioni. La prima delle precedenti costruzioni per inferenza è espressa in forma corrispondente a un enunciato in Prolog. La differenza essenziale tra il Prolog e gli altri linguaggi dichiarativi e i linguaggi procedurali tradizionali, quali il Pascal, è che gli asserti logici in un programma Prolog sono validi contemporaneamente. Non è previsto che vengano applicati sequenzialmente, e molti dei problemi associati alla programmazione logica scaturiscono dalla limitatezza degli attuali processori sequenziali. Implicito nell'elaborazione inferenziale e nel software di programmazione logica che fa da supporto è un grado molto alto di elaborazione parallela. Il Prolog e altri potenziali linguaggi dei computer della quinta generazione sono trattati nel capitolo 7; i sistemi intelligenti basati sulla conoscenza sono l'argomento del capitolo 8.
Per ottenere i numeri sopra riportati per le prestazioni dall'elaborazione inferenziale, è necessario che i computer della quinta generazione siano forniti di un particolare meccanismo hardware di elaborazione logica, nella figura 4.1 descritto come "elaboratore inferenziale e di risoluzione di problemi". Esso richiede del supporto hardware diretto per gli enunciati logici del tipo discusso sopra, basato su un meccanismo che possa gestire le grandi dimensioni e la complessità di un problema di inferenza "reale". È probabile che questi problemi richiedano milioni di enunciati inferenziali, molti dei quali devono essere interpretati in parallelo.

Una valida architettura hardware per questa situazione è quella nota col nome di *architettura a flusso di dati*, in cui numerose unità di elaborazione discrete (sequenziali) sono connesse in rete. Ogni unità di elaborazione si attiva non appena riceve un insieme completo di dati su cui operare, e trasmette i suoi risultati alla rete non appena l'operazione è completa. In questo modo i dati fluiscono attraverso la rete, che a sua volta è strutturata in modo da riflettere il grado di parallelismo pertinente alla particolare applicazione. (Per ulteriori dettagli si vedano i paragrafi 5.4 e 5.5.)
Per finire con un'ultima considerazione, alla base del suo elaboratore inferenziale e di risoluzione dei problemi, il modello JIPDEC include un "meccanismo di Von Neumann innovativo". Come abbiamo visto nel paragrafo 1.4, le caratteristiche essenziali di un elaboratore di Von Neumann sono la codifica simbolica di dati e istruzioni, l'assenza di una distinzione fondamentale tra dati e istruzioni, e l'elaborazione sequenziale delle istruzioni per mezzo di un numero ridotto di registri di elaborazione. Tutti i computer costruiti finora sono variazioni su questo tema, ed è ormai possibile realizzare una rete a flusso di dati utilizzando degli elaboratori di Von Neumann. Comunque, c'è spazio per apportare ulteriori innovazioni a questo concetto di base, e può verificarsi anche che l'alto livello di parallelismo necessario in tutti i campi dell'elaborazione della quinta generazione richieda qualcosa di sostanzialmente diverso. (Si veda il paragrafo 5.9.)

4.4 Interfacce utente intelligenti

L'espressione "*user-friendly*", ovvero "amichevole verso l'utente" è riuscita ad attirare, soddisfare o deludere gli utenti delle prime quattro generazioni di computer, ma esprime un concetto troppo superficiale per la quinta generazione. Ciò che è semplice per un ingegnere di aereodinamica in un istituto di ricerca non è altrettanto semplice per un paramedico in Etiopia. Per la quinta generazione la situazione deve essere capovolta: è il computer, infatti, a dover fare i conti con la natura e il modo di pensare del suo utente, e non viceversa. L'interazione tra persona e computer deve essere un'estensione naturale dell'approccio umano al compito, ossia: "Risptetto ai sistemi tradizionali, l'interfaccia uomo-macchina sarà più vicina al sistema umano" (JIPDEC, 1981). Per quanto l'utente possa essere un esperto in un settore quale la medicina o la geologia, si può presupporre che abbia solo un

livello elementare di cultura informatica.
Ci potrebbe essere ancora un posto per l'interazione tra l'utente e il suo computer basata sull'introduzione di caratteri per mezzo di tastiere tradizionali e di schermi video, ma, probabilmente, sarà limitata alle aree applicative più appropriate, quali l'introduzione o l'aggiornamento delle basi di conoscenza utilizzate in situazioni più dinamiche. La tendenza attuale che prevede l'impiego di icone e mouse per il controllo dei programmi potrà svilupparsi ulteriormente nelle applicazioni della quinta generazione. Comunque, due importanti canali di comunicazione tra i computer della quinta generazione e i loro utenti saranno, quasi sicuramente, il linguaggio naturale e gli schermi video. Come già si è detto nei paragrafi 2.7 e 2.8, in passato i progressi in questi due settori sono stati ostacolati dalla presenza di problemi fondamentali irrisolti.
Nel caso del linguaggio naturale, la meta operativa è riuscire ad automatizzare sottoinsiemi di un linguaggio abbastanza estesi da essere utili in particolari situazioni. Per ritornare all'esempio sopra citato, un paramedico in Etiopia utilizza solo una parte ridotta di tutto il vocabolario medico, i cui termini hanno un significato ben preciso, e il contesto generale dell'interazione è noto al computer. In modo analogo, l'elaborazione delle immagini verrà utilizzata in circostanze relativamente ben precise, quali impianti industriali di montaggio, operazioni chirurgiche (sonde negli organi che restituiscono immagini attraverso fibre ottiche) e manutenzione dei satelliti nello spazio. In ciascuno di questi casi la classe di oggetti da riconoscere è sufficientemente ristretta da essere inclusa nella gamma delle attuali attività di ricerca e sviluppo. (Per maggiori dettagli si veda il capitolo 9.)
Più in generale, la psicologia dell'interazione tra una persona e un computer sta diventando sempre più oggetto di studi volti a fornire una solida base per la progettazione delle interfacce utente intelligenti. Qualunque sia la tecnica scelta per una particolare applicazione, sarà necessario il supporto di particolari sistemi software e hardware che interagiscano a vari livelli con i sistemi di elaborazione inferenziale del computer.

4.5 Hardware e software della quinta generazione

È giunto il momento di dire qualcosa sui sistemi hardware e software della quinta generazione. Tutto indica che, nonostante la frattura concettuale rispetto ai com-

puter delle generazioni precedenti, l'hardware dei computer della quinta generazione sarà basato sull'altissima integrazione di componenti a semiconduttori. Il grado di integrazione sarà di almeno due ordini di grandezza superiore all'attuale: centinaia di migliaia di elementi discreti per chip. Motivo di ricerca e sviluppo è stabilire se il silicio rimarrà l'ingrediente di base o se sarà sostituito da un altro semiconduttore, per esempio l'arsenurio di gallio. (Per maggiori dettagli si veda il capitolo 5.)
La maggior parte dei programmi odierni di elaborazione dati sono dell'ordine di decine o centinaia di migliaia di righe di codice sorgente in un linguaggio procedurale come il Cobol o l'Ada. È certo che in un computer della quinta generazione ciascuno degli strati di software richiederà milioni di righe di codice sorgente, in linguaggi basati sulla logica come il Prolog o linguaggi a flusso di dati come l'Occam. Inoltre, dato che la struttura hardware di ciascun tipo di computer della quinta generazione sarà adattata alla sua particolare area di applicazione più di quanto non avvenga oggi, sarà impossibile, nella maggior parte dei casi, sviluppare il software sull'unità di elaborazione su cui dovrà effettivamente girare. Se a questo aggiungiamo le dimensioni e la complessità dei programmi, per preparare pacchetti software per la quinta generazione saranno necessarie tecniche avanzate di ingegneria del software che utilizzino ambienti di sviluppo fatti su misura, nei quali potranno essere presenti strumenti per dimostrare la correttezza dei moduli di programma mediante tecniche formali, come se si trattasse di teoremi matematici. Gli aspetti di ingegneria del software dei computer della quinta generazione sono trattati nel capitolo 6.

4.6 Conclusione

I computer sono già quanto di più complesso si sia mai prodotto, ma le prospettive per i computer della quinta generazione sono tali da determinare un notevole aumento del grado di complessità. La sfida lanciata ai gruppi di ricercatori dei vari paesi è riuscire a gestire questa complessità e creare, in breve tempo, un aggregato di elementi funzionali ciascuno dei quali si trova al limite delle ricerche attuali, o decisamente al di là. Alla fine di questo cammino sta la meta enunciata nel rapporto originale (JIPDEC, 1981): "l'intelligenza verrà aumentata tanto da avvicinarsi a quella umana".

5

Struttura hardware della quinta generazione

Gli sviluppi più sorprendenti compiuti nel settore informatico negli ultimi anni riguardano l'hardware. Dalla messa a punto del primo microprocessore del 1972, il numero di elementi funzionali per chip è aumentato costantemente. Le memorie monochip sono passate rapidamente dagli 8 k ai 16 k ai 64 k fino agli attuali 256 k, e sono già stati realizzati prototipi di chip con 1 megabyte di memoria. Il prezzo per unità di prestazione o di memorizzazione continua a diminuire e l'affidabilità di questi complessi dispositivi a stato solido è estremamente alta. Ma, confrontati con i requisiti necessari all'hardware della quinta generazione, questi sviluppi recenti sembrano insignificanti.

Il maggior progresso finora compiuto nello sviluppo dell'hardware è stato riuscire a compattare sempre più nel silicio le architetture tradizionali di elaborazione e memoria. Questo ha determinato un aumento del "divario semantico" tra l'alto livello delle funzioni che si richiedono a un computer — come l'elaborazione per la risoluzione di problemi in linguaggi come Pascal, Prolog o Ada — e il basso livello della sua architettura. Una delle motivazioni alla base degli sviluppi dell'hardware della quinta generazione è ridurre il divario semantico ossia sviluppare "funzioni di efficace supporto da implementare a livello dell'architettura dell'elaboratore per la realizzazione di sistemi informativi di elaborazione della conoscenza" (Aiso, 1982). Pertanto, oltre a essere sostanzialmente più potente, l'hardware della quinta generazione sarà basato su architetture concettualmente del

tutto diverse da quelle che si trovano realizzate sulle macchine attuali.
Un metodo di affrontare il problema è considerare gli aspetti hardware e software di un nuovo sistema di elaborazione come un'unica struttura e permettere che la "reciproca interazione fra algoritmi, architettura e tecnologia determini una struttura equilibrata in grado di soddisfare gli obiettivi del sistema" (Allen, 1982). Questo punto di vista deve essere adottato tenendo conto che i requisiti della struttura generale di un sistema della quinta generazione sono un assemblaggio interattivo contenente un elaboratore della base di conoscenza, un elaboratore inferenziale e un'interfaccia utente intelligente. Le strutture equilibrate che forniscano le prestazioni richieste ai tre sottosistemi potranno rivelarsi non molto simili. Comunque, si nutre la convinzione che ci debba essere un unico concetto di base dell'architettura dei computer della quinta generazione, e forse un linguaggio "nucleo" comune (a grandi linee l'equivalente del linguaggio macchina dei sistemi attuali) (Treleaven, 1982).

5.1 Altissima integrazione

Per quanto l'architettura degli elaboratori della quinta generazione sia completamente diversa da quella degli elaboratori di oggi e i requisiti di memoria e dell'elaborazione siano di gran lunga superiori, la tecnologia fondamentale dell'hardware rimane invariata: altissima integrazione dei componenti a semiconduttori. Si stanno svolgendo ricerche sull'arsenurio di gallio quale alternativa al silicio, e su nuovi modi di raffreddare chip di elevate dimensioni e densamente impaccati. Tutto questo, comunque, non è che un'estrapolazione della tendenza del momento: impaccare sempre più componenti nella stessa piccola area di silicio.
La complessità dei chip ad altissima integrazione più avanzati è chiaramente illustrata in un'analogia tracciata da Charles Seitz. Se i primi circuiti integrati fossero tanto complessi quanto la rete stradale di un piccolo distretto, "la tecnologia del quarto di micrometro potrebbe essere in grado di produrre dei chip tanto intricati quanto una rete stradale urbana che coprisse l'intero continente nordamericano" (Seitz, 1980). La costante riduzione delle dimensioni degli elementi dei chip determinerebbe enormi problemi d'ingegneria e di controllo della qualità ad ogni stadio di fabbricazione. Uno degli ostacoli da superare è la larghezza di una pista conduttrice che attualmente è dell'ordine di mezzo micrometro, ma che probabil-

mente sarà ridotta a un quarto di micrometro nel corso del programma della quinta generazione di computer. Per soddisfare queste esigenze, la precisione nei diversi stadi di fabbricazione di un chip dovrà aumentare continuamente, e si dovrà incorporare nella struttura del chip un certo grado di ridondanza per compensare inevitabili imperfezioni. Ma il vincolo principale è la velocità della luce, ossia la velocità con cui i segnali elettrici si propagano lungo un conduttore. Per ottenere prestazioni nell'ordine dei nanosecondi, tutti gli elementi di elaborazione di un computer devono essere contenuti entro un volume di 30 cm cubi (la luce percorre circa 30 cm in un nanosecondo).

Una tecnica per l'altissima integrazione con una storia piuttosto movimentata è l'integrazione sulla scala del wafer. Il processo di fabbricazione di un chip tradizionale è formare qualche centinaio di singoli chip identici su un disco circolare (un *wafer*) di di silicio cristallino. Il passo successivo, apparentemente ovvio, è utilizzare l'intero wafer di silicio per un unico chip, di circa 5 cm quadrati. Questo metodo venne tentato senza successo dalla Trilogy Corporation di Gene Amdahl, all'inizio degli anni ottanta. Il problema riguarda il controllo di qualità: i chip tradizionali vengono messi alla prova prima di essere separati dal loro substrato, e quelli difettosi vengono scartati. In genere il prodotto selezionato varia notevolmente, ma raramente supera il 90%. Per essere utilizzabile, un chip con integrazione a wafer di silicio deve essere perfetto, o vicinissimo alla perfezione, perché la ridondanza nella sua struttura sia sufficiente. Non si sono, comunque, ancora raggiunti gli standard di produzione richiesti per un potenziale di parecchi milioni di elementi funzionali e una complessità superiore di un ordine di grandezza ai più grandi chip tradizionali.

Supponendo di aver risolto i problemi di fabbricazione, il problema principale è l'architettura del chip: come gestire la molteplicità e complessità che possono essere presenti in schiere con milioni di elementi funzionali. Molti dei metodi più recenti (paragrafo 5.8) puntano a ottenere il massimo da ampie combinazioni di elementi semplici, con percorsi di controllo brevi e regolari (Mead e Conway, 1980). La novità è data dalla natura degli elementi di elaborazione e dal grado di parallelismo incorporato.

5.2 Elaborazione parallela

Tutti gli operatori del settore si trovano d'accordo sul fatto che l'elemento che contraddistingue l'architettura dei computer della quinta generazione è un grado di parallelismo maggiore di quello incorporato negli attuali computer. Probabilmente vi saranno vari livelli di funzionamento in parallelo: dall'accoppiamento stretto dei circuiti di elaborazione che riflettono il parallelismo insito nelle operazioni di elaborazione inferenziale o della base di conoscenza, all'accoppiamento lasco tra i vari sottosistemi in un computer della quinta generazione e all'elaborazione distribuita attraverso reti locali o geografiche.
Fino ad ora negli elaboratori sono stati realizzati due tipi di parallelismo in accoppiamento stretto: le *matrici di elaborazione* (processing array) e i *pipeline*. Le matrici di elaborazione sono vettori di unità di elaborazione identiche che funzionano in modo sincrono per eseguire operazioni identiche su matrici di dati. I pipeline vengono utilizzati per operazioni a più stadi come la moltiplicazione in virgola mobile, in cui ogni elemento del pipeline esegue un ciclo di elaborazione e passa il suo risultato intermedio all'elemento successivo. Le operazioni su insiemi consecutivi di dati possono svolgersi a cicli alterni. L'elaborazione parallela di questo tipo, nota col nome di elaborazione parallela "regolare" (Gurd, 1982), sarà certamente presente nei computer della quinta generazione. Va notato, comunque, che uno dei principali argomenti di ricerca sono i meccanismi necessari per gestire elaborazioni in parallelo irregolari. È opportuno, ora, parlare ulteriormente di tre metodi: flusso di controllo parallelo, flusso di dati e riduzione di grafi.

5.3 Flusso di controllo parallelo

Solitamente, ogni ciclo di elaborazione viene eseguito in sequenza sotto il controllo di un contatore che, all'interno di ogni singolo programma, determina qual è la successiva operazione di basso livello da eseguire. Il flusso di controllo è implicito nella struttura del programma. Ad esempio, il modulo di controllo di un programma in un linguaggio tipo Pascal per l'elaborazione degli stipendi potrebbe essere come il seguente:

```
begin
    leggi (record_salario);
    calcola_stipendio_lordo (record_salario);
    calcola_imposta_reddito (record_salario);
    calcola_previdenza_sociale (record_salario);
    calcola_stipendio_netto (record_salario);
    scrivi (record_salario)
end
```

Ogni istruzione nel modulo è una chiamata a una procedura di elaborazione più dettagliata. Risulta che il calcolo dell'imposta sul reddito e quello della previdenza sociale sono indipendenti l'uno dall'altro: pertanto, se fossero disponibili un sistema di computer paralleli e dei linguaggi di programmazione in parallelo, il modulo di controllo potrebbe essere scritto nel seguente modo:

```
begin
    leggi (record_salario);
    calcola_stipendio_lordo (record_salario);
    parallelo
        calcola_imposta_reddito (record_salario);
        calcola_previdenza_sociale (record_salario)
    end;
        calcola_stipendio_netto (record_salario);
        scrivi (record_salario)
    end
```

Le procedure di elaborazione dell'imposta e della previdenza sociale vengono chiamate contemporaneamente. Esse vengono eseguite in parallelo e il modulo di controllo aspetta che entrambe siano terminate prima di continuare. I linguaggi di programmazione come il Pascal concorrente e l'Ada (paragrafo 7.3) possono gestire operazioni di questo tipo, ma resta da vedere se questo metodo, che differisce leggermente dall'elaborazione sequenziale tradizionale, sarà adatto a soddisfare i requisiti di base dell'architettura della quinta generazione.

5.4 Architettura a flusso di dati

L'architettura a flusso di dati è, per più motivi, una delle configurazioni più promettenti per i sottosistemi di elaborazione inferenziale di un computer della quinta generazione. Essa, infatti, può gestire il parallelismo in accoppiamento stretto sia irregolare che regolare, è flessibile ed estensibile, ha un elevato potenziale di trattamento dei dati e riflette, a livello dell'hardware, il tipo di parallelismo insito nell'elaborazione inferenziale (Tanaka, 1982). L'approccio a flusso di dati è fondamentale nel programma giapponese della quinta generazione e sta diventando sempre più importante anche nei progetti del Regno Unito.
Per spiegare con un esempio alcuni dei concetti di un'architettura a flusso di dati consideriamo nuovamente l'enunciato di tipo Prolog:

Se sintomo-di (*x*, *y*) e presenta (*z*, *x*) allora soffre di (*z*, *y*).

Supponendo che un computer del giorno d'oggi sia in grado di valutare un predicato o un'operazione binaria booleana in un passo procedurale, un compilatore sequenziale potrebbe tradurre l'esempio per mezzo delle variabili intermedie booleane A e B, come illustrato:

A = sintomo-di (*x*, *y*)
B = presenta (*z*, *x*)
soffre-di (*z*, *y*) = A e B.

In un computer tradizionale, le risultanti tre istruzioni in linguaggio macchina verrebbero eseguite una di seguito all'altra. Comunque, un altro modo per rappresentare la struttura di questo enunciato è per mezzo di un grafo, come nella figura 5.1. (Una versione equivalente di questa struttura a grafo verrebbe prodotta dal compilatore quale fase intermedia prima di ottenere le tre istruzioni in linguaggio macchina sopra riportate.) Si può interpretare il flusso di dati di questa struttura per mezzo di tre circuiti di elaborazione, come illustrato nella figura 5.2, due dei quali operino in parallelo.
Questo semplice esempio illustra l'idea centrale di un'architettura a flusso di dati: viene preparata una rete di circuiti di elaborazione che rispecchi la strutttura logica dell'operazione da svolgere, e i singoli dati scorrono tra gli elementi.
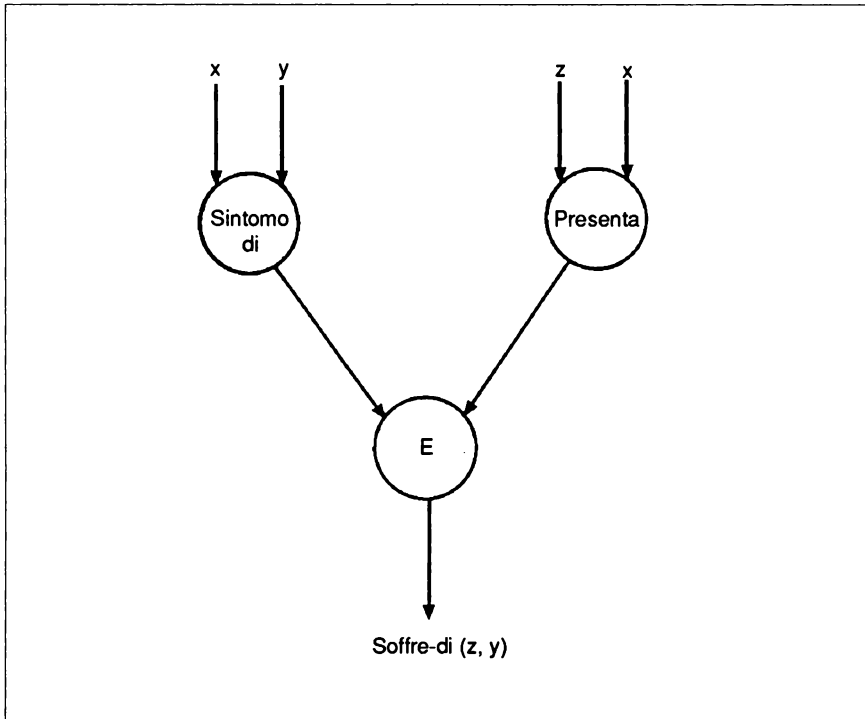
Fig. 5.1 Diagramma di inferenze multiple.

Ogni elemento opera con un suo ritmo e aspetta di avere un insieme completo di input intermedi prima di "attivarsi". Esistono due tecniche di controllo di questo tipo di rete: secondo l'approccio dei dati attivi ogni elemento aspetta passivamente che i dati arrivino, mentre in regime di dati passivi è il singolo elemento che, quando è pronto, richiede i dati "a fonte" (Sharp, 1985).
In generale tre sono i requisiti di un elaboratore o di un sottosistema di elaboratori a flusso: memorizzare le rappresentazioni dei grafi dei programmi, realizzare un qualche tipo di elemento di dati che fluisca attraverso i grafi e fornire strumenti adeguati per elaborare le istruzioni. Ciascuno di questi tre requisiti presenta problemi, alcuni piuttosto grossi. In pratica, ogni grafo di programma conterrà centinaia di migliaia, se non milioni, di archi e nodi e non sarà pertanto sempre riducibile alla semplice struttura ad albero illustrata nella figura 5.2.
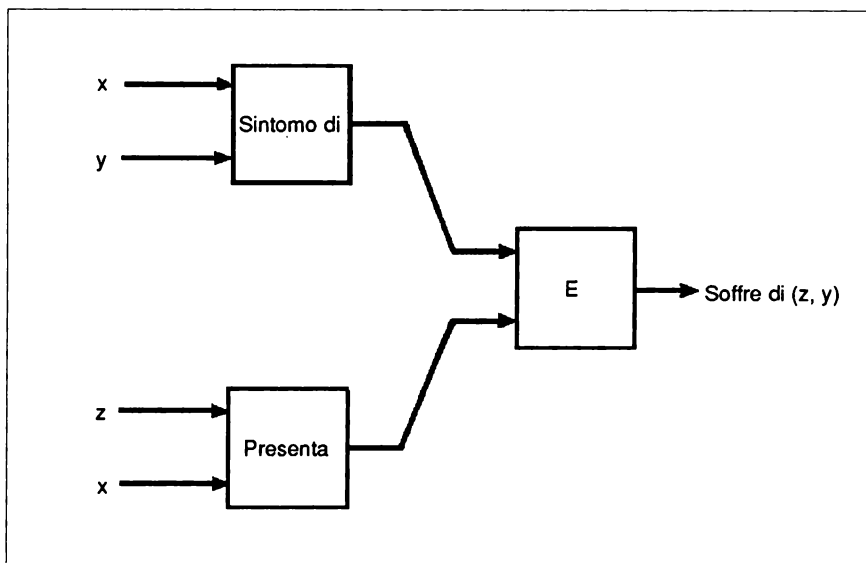
Fig. 5.2 Rete a flusso per la figura 5.1.

Inoltre, se, come probabile, il programma dovesse contenere definizioni ricorsive, alcune parti delle strutture dovranno essere contenute in se stesse. Si consideri, ad esempio, la frase:

Se figlio-di(*x*, *y*) o figlio-di(*x*, *z*) e discendente-di (*z*, *y*)
allora discendente-di(*x*, *y*)

(Questo significa che una persona *x* è discendente di una persona *y* sia se *x* è figlio di *y* sia se esiste una persona *z* tale che *x* sia figlio di *z* e *z* discendente di *y*.) Come illustrato nella figura 5.3, durante l'elaborazione il grafo di questa inferenza deve duplicare se stesso ripetutamente. La maggior parte dei dati elaborati nei sistemi basati sulla conoscenza non sono costituiti da singoli elementi, ma da grosse strutture. Queste, se fossero fatte passare interamente attraverso una rete a flusso, causerebbero degli inaccettabili appesantimenti. Per risolvere questo problema si è pensato di utilizzare dei puntatori alle strutture di dati nella rete a flusso, e di accedere alle strutture in memoria solo quando siano necessarie.
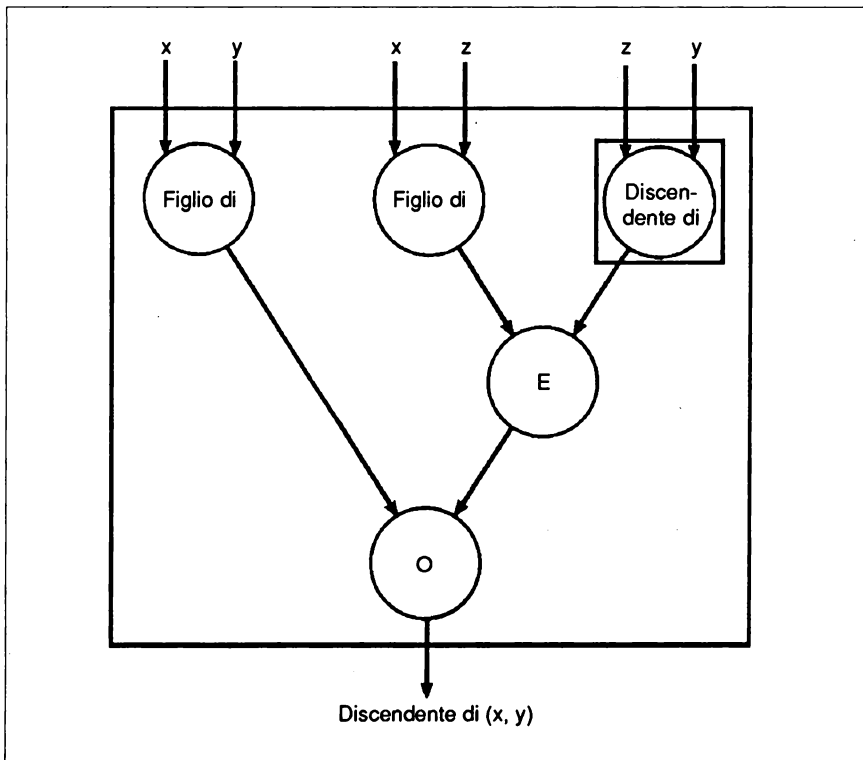
Fig. 5.3 Diagramma ripetitivo: il riquadro grande deve essere riprodotto ripetutamente all'interno di quello piccolo.

Per superare queste difficoltà si stanno seguendo tre linee di ricerca. La prima è considerare un compito a flusso come fissato in compilazione, e non ammettere il codice che richiama se stesso.
Questo metodo statico è illustrato nella figura 5.4, che utilizza una rete di unità di elaborazione binaria, ciascuna con due canali alternativi per l'uscita. Il metodo dinamico elude il problema del codice ricorsivo consentendo la duplicazione di porzioni della rete durante l'esecuzione. Oltre ad essere semplice, questo metodo ha il vantaggio di poter essere realizzato tanto più facilmente quanto più deboli sono i vincoli dell'hardware. La figura 5.5 illustra una configurazione possibile che utilizzi questa tecnica (secondo Tanaka, 1982).

La linea di sviluppo che sembra più promettente a breve termine è il sistema a indicatori, variazioni del quale sono in via di sviluppo al MIT e all'Università di Manchester. Ogni dato che attraversa la rete porta con sé un identificatore che specifica le sue caratteristiche (può essere, per esempio, un puntatore a una grande struttura di dati residente nella memoria di massa) e la sua posizione nel programma. L'identificatore permette di accoppiare il dato con l'istruzione adatta alla sua elaborazione. Gli identificatori individuano anche il livello di ricorsione qualora venga usato codice ricorsivo.
Nella figura 5.6 è illustrato un nodo di un sistema a flusso che utilizza questa tecnica (secondo Gurd,1982).
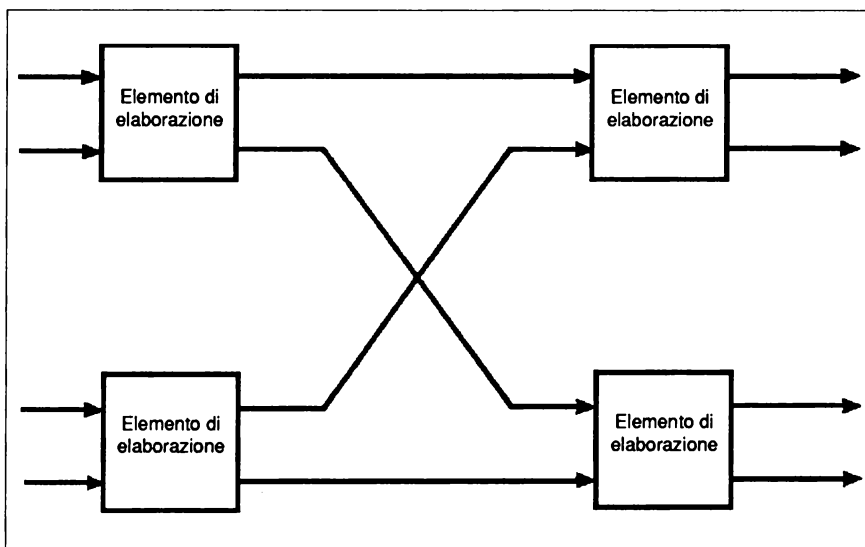
Fig. 5.4 Rete a flusso statica (rete a delta).
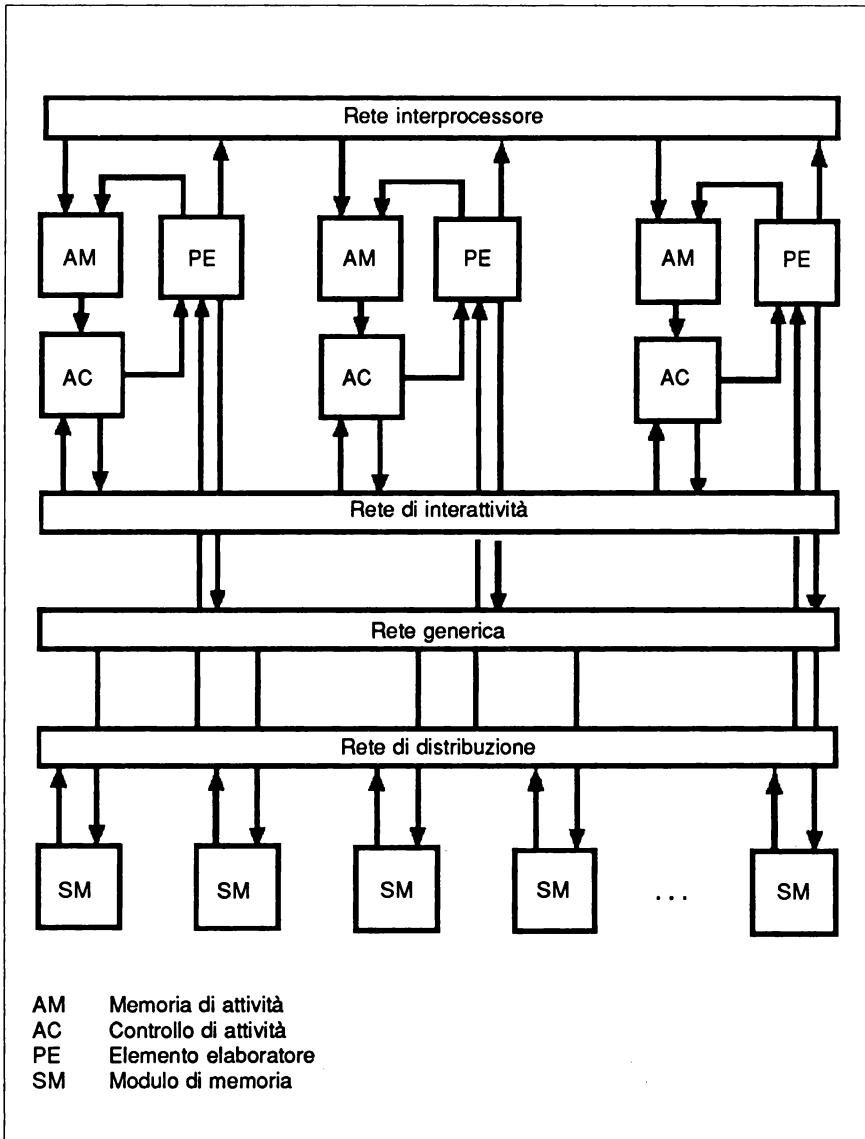
Fig. 5.5 Architettura a flusso dinamica.
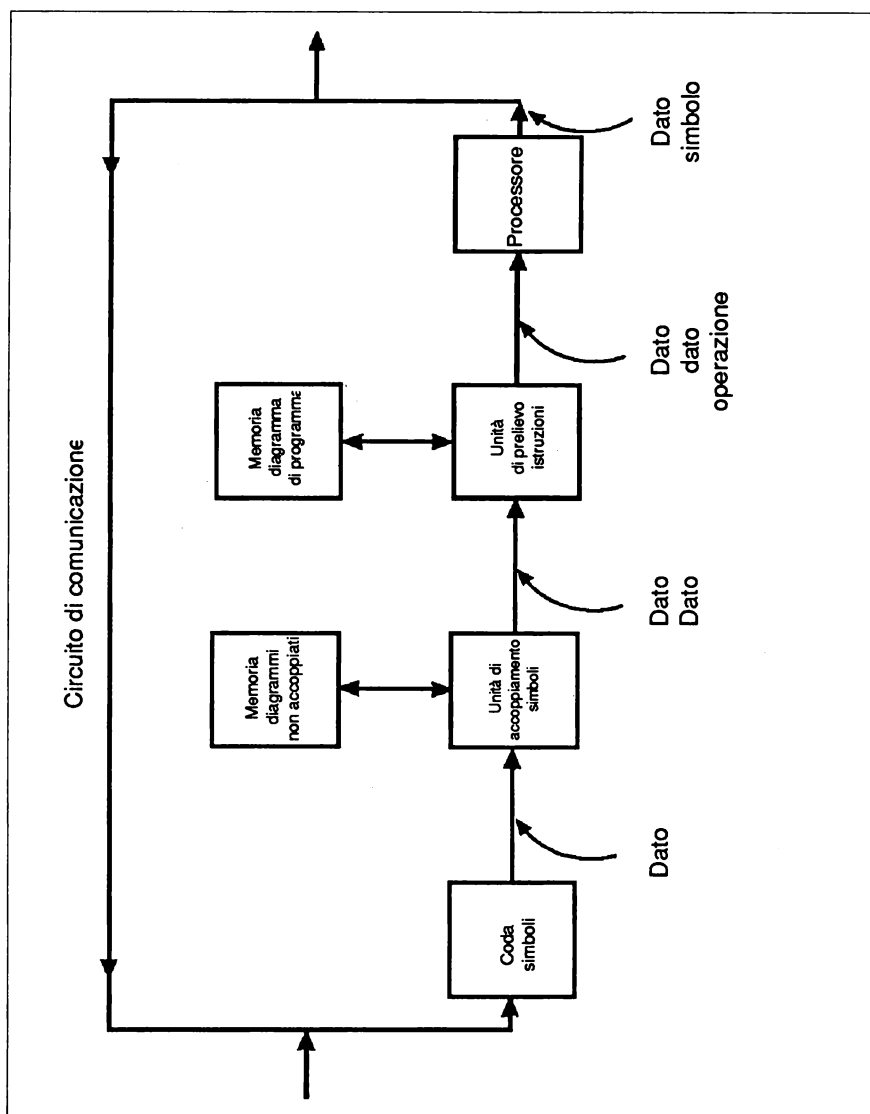
Fig. 5.6 Architettura a fiusso con identificatori.

5.5 Architettura di riduzione dei grafi

Una variante del metodo a flusso di dati consiste nel valutare le funzioni operando direttamente sulle loro rappresentazioni grafiche. Via via chevengono valutate, le varie parti del grafo sono sostituite dai loro risultati intermedi. In queto modo, continuando con la valutazione, il diagramma viene ridotto. Per esempio, si consideri ancora l'enunciato:

Se figlio-di (*x*, *y*) o figlio-di(*x*, *z*) e discendente-di
(*z*, *y*) allora discendente-di (*x*, *y*)

Date le relazioni:

figlio-di (Elisabetta, Giorgio)
figlio-di (Carlo, Elisabetta)
figlio-di (Guglielmo, Carlo)

il calcolo del discendente-di (Guglielmo, Giorgio) si sviluppa nel grafo illustrato nella figura 5.7. La valutazione di ciascuno dei nodi inferiori (che diventa una ricerca per vedere se tale nodo è presente nelle relazioni date), può essere svolta in parallelo. I risultati booleani intermedi vengono quindi fatti risalire lungo il grafo via via che questo viene ridotto, fino a quando ne esce un singolo risultato. I vari passaggi sono illustrati nella figura 5.8.

5.6 Alice

All'Imperial College di Londra si sta mettendo a punto un computer che incorpora la riduzione dei grafi direttamente nella sua struttura hardware (Darlington e Reeve, 1981; Cripps, Field e Reeve, 1985). Chiamata Alice (che sta per Applicative Language Idealised Computing Engine, motore di calcolo idealizzato per linguaggi applicativi), costituisce una delle prime architetture complete di computer della quinta generazione prodotte dai vari gruppi di sviluppo. È progettata per essere programmata nel linguaggio applicativo Hope (paragrafi 7.7 e 7.8) ma è compatibile anche con linguaggi dichiarativi come Prolog (paragrafi 7.5 e 7.6).
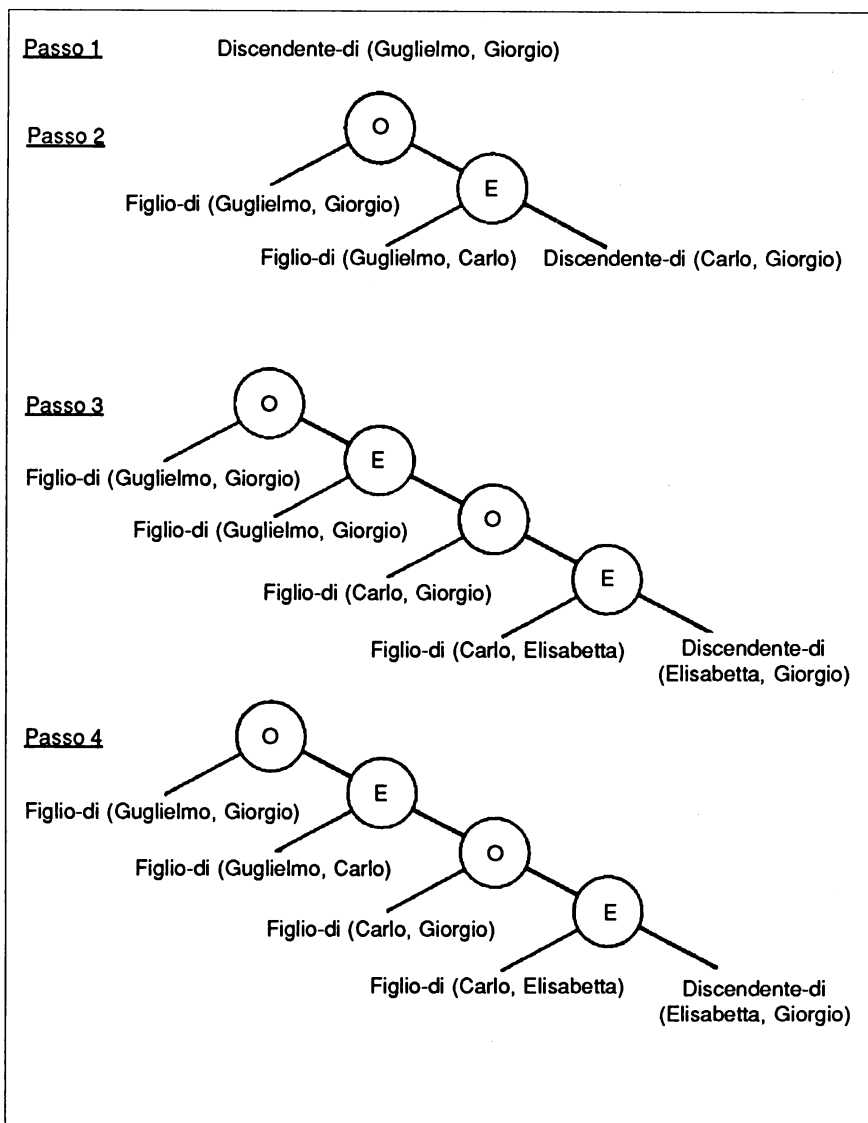
Fig. 5.7 Grafo di relazione di discendenza: scomposizione.
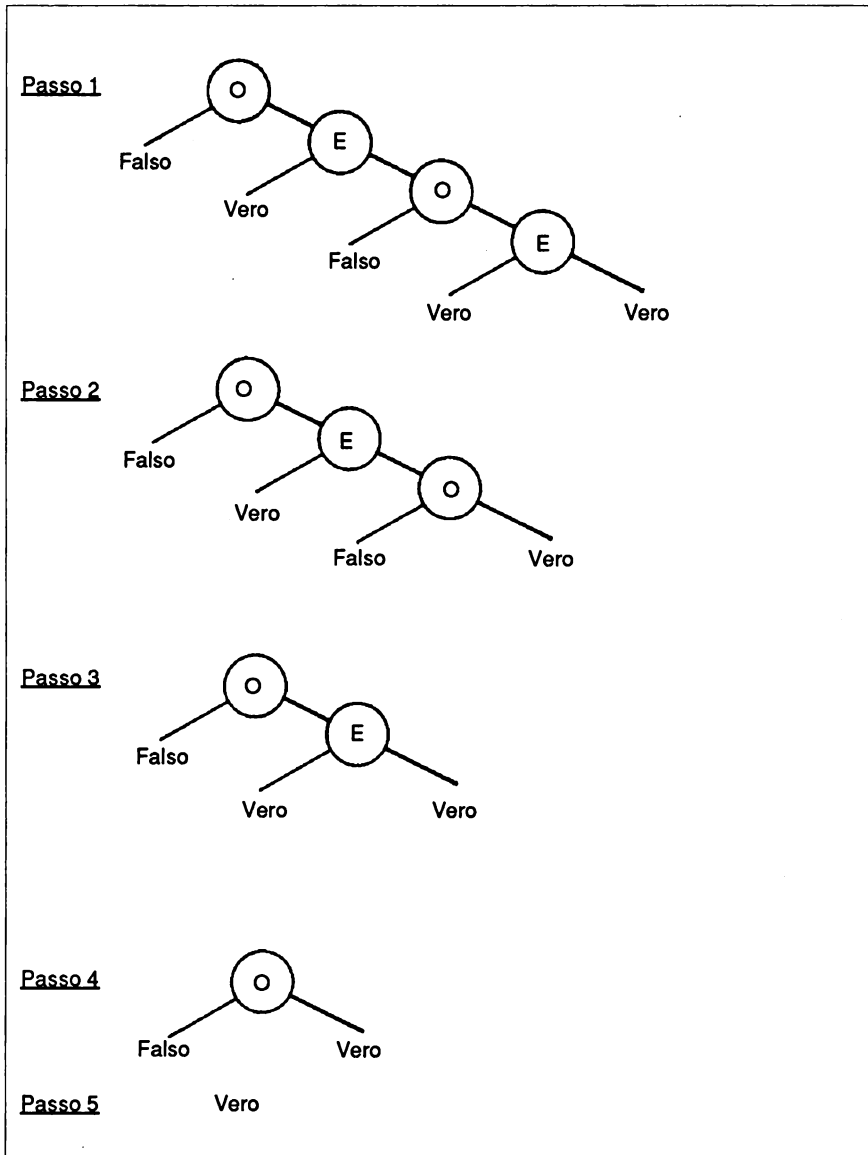
Fig. 5.8 Grafo di relazione di discendenza: riduzione.

Dato un certo compito di elaborazione nella forma di grafo di funzione, Alice prima disegna il grafo sostituendo a ogni nodo la definizione della funzione corrispondente, fino a quando il grafo è completamente calcolabile. Quindi riduce il grafo calcolando il valore della funzione in ogni nodo e sostituendo questo valore nei nodi superiori. Molti degli stadi dei processi di creazione di riduzione possono svolgersi in parallelo; la struttura di Alice fa sì che le operazioni parallele vengano eseguite senza che il programma dia alcuna istruzione esplicita.

All'interno di Alice ogni nodo in un grafo di programma è rappresentato come un pacchetto. Un pacchetto è composto da un campo identificatore, una funzione o un campo operatore, e uno o più campi argomento che possono essere valori dati o riferimenti ad altri pacchetti. Ci sono anche campi di controllo che l'elaboratore utilizza durante le sue operazioni. I pacchetti corrispondenti all'esempio sopra citato sono illustrati nella figura 5.9.

La creazione del grafo viene effettuata per mezzo di un processo di messa in corrispondenza degli elementi: gli argomenti di una funzione sono messi in corrispondenza con la definizione del pacchetto, e in questo modo il pacchetto viene sostituito da un'ulteriore funzione o operatore. La riduzione procede esaminando i pacchetti operatore: qualsiasi pacchetto con tutti gli argomenti espressi come valori (e non come riferimenti ad altri pacchetti) viene valutato e sostituito da un pacchetto contenente il valore risultante. Questo risultato viene quindi sostituito in tutti i pacchetti in cui sia necessario utilizzando allo scopo i vari campi di controllo del pacchetto.

La struttura generale di un computer Alice è illustrata nella figura 5.10. Consiste di una grossa memoria segmentata che fa da pool del pacchetto, e da un certo numero di agenti di elaborazione. Le unità di elaborazione e i segmenti di memoria sono connessi per mezzo di una rete di commutazione ad alta velocità che consente a ogni unità di elaborazione di accedere a qualsiasi segmento della memoria in un tempo ridottissimo rispetto agli altri metodi di accesso.

La configurazione scelta è una rete a delta, comprendente molti circuiti di commutazione elementari con quattro entrate e quattro uscite in una schiera regolare (in figura 5.4 è illustrata una rete a delta di elementi con due entrate e due uscite.) La rete opera in modo asincrono, cosicché ogni richiesta di un pacchetto viene trasmessa attraverso i circuiti di commutazione il più rapidamente possibile, e il pacchetto viene restituito all'unità di elaborazione non appena viene aperto il percorso di accesso.

SCOMPOSIZIONE

Passo				
Passo 1	P1	Discendente-di	Guglielmo	Giorgio
Passo 2	P1	O	P2	P3
	P2	Figlio-di	Guglielmo	Giorgio
	P3	E	P4	P5
	P4	Figlio-di	Guglielmo	Carlo
	P5	Discendente-di	Carlo	Giorgio
	P5	O	P6	P7
Passo 3	P6	Figlio-di	Carlo	Giorgio
	P7	E	P8	P9
	P8	Figlio-di	Carlo	Elisabetta
	P9	Discendente-di	Elisabetta	Giorgio
Passo 4	P9	Figlio-di	Elisabetta	Giorgio

RIDUZIONE

Passo		
Passo 1	P2	FALSO
	P4	VERO
	P6	FALSO
	P8	VERO
	P9	VERO
Passo 2	P7	VERO
Passo 3	P5	VERO
Passo 4	P3	VERO
Passo 5	P2	VERO

Fig. 5.9 Pacchetti Alice per relazione di discendenza.

Anche il collegamento delle singole unità di elaborazione costituisce una rete di distribuzione a banda stretta, contenente sia gli indirizzi dei pacchetti elaborabili che quelli dei pacchetti vuoti. Questa rete comprende circuiti di elaborazione semplici che trasferiscono gli indirizzi da un agente di elaborazione all'altro, per smaltire la mole di lavoro in attesa presso ogni unità di elaborazione. Alice utilizza il transputer Inmos (paragrafo 5.7) quale elemento di elaborazione di base: ogni agente principale contiene un certo numero di transputer, e ulteriori transputer forniscono l'intelligenza alla rete di distribuzione.
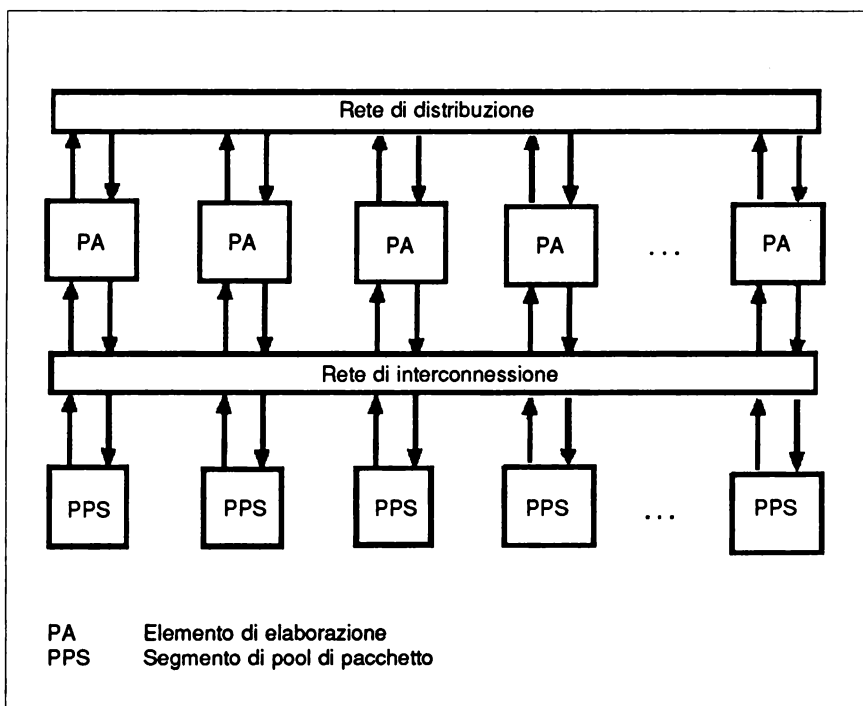
Fig. 5.10 Alice: struttura generale.

5.7 Il transputer Inmos

Il transputer Inmos è una delle possibili configurazioni di un elemento di elaborazione di un computer della quinta generazione. Questo transputer, che ha destato l'attenzione di tutti i gruppi che si occupano dello sviluppo della quinta generazione, è strutturato come un elemento di elaborazione *single-chip* per architetture parallele (Smith, 1983). È provvisto di memoria incorporata, con dispositivi per DMA ad alta velocità (accesso diretto alla memoria per canali di input ed output, senza passare attraverso l'unità di elaborazione), e registri di ricezione e trasmissione per il trasferimento dei dati tra i transputer. La sua unica unità di elaborazione sequenziale ha un insieme ridotto di istruzioni che funziona alla massima velocità, fornendo ai prototipi un tempo di esecuzione di un ciclo di istruzione pari a 50 nanosecondi. Il transputer ha un'elevatissima capacità di trattamento dei dati, anche se la velocità di elaborazione non è altrettanto elevata.
Il transputer è pensato per la programmazione diretta in Occam (paragarafo 7.4) ed è destinato a essere incorporato in un'architettura distribuita, con singoli transputer connessi per mezzo di reti locali ad altissima velocità. Come tale, costituisce un blocco ideale per la costruzione di molti componenti di un sistema di computer della quinta generazione.

5.8 Architetture non di Von Neumann

A partire dal 1946 i computer sono sempre stati basati su unità di elaborazione con un unico insieme di registri contenenti le istruzioni e i dati in elaborazione, e seguono un ciclo fisso di operazioni per prelevare ed eseguire ogni istruzione macchina (paragrafo 1.4). Finora, questa tradizionale architettura di Von Neumann, in virtù della sua semplicità e flessibilità, ha soddisfatto le esigenze della tecnologia informatica, ma il collo di bottiglia che si crea in presenza di un unico ciclo di elaborazione e di un unico insieme di registri è troppo limitativo per la quinta generazione. Comunque, va evitata anche la complessità potenziale delle matrici ad altissima integrazione che non vengono disposte secondo configurazioni generali semplici.
Le principali impostazioni di nuove architetture dei chip prevede l'abbandono delle strutture non specializzate dei processori tradizionali e la messa a punto di chip che

siano efficienti all'interno di una gamma vasta, ma allo stesso tempo specifica, di operazioni, quali operazioni di ricerca o di confronto. A questo scopo, i requisiti sono i seguenti: pochi semplici elementi di elaborazione elementare replicati molte volte sul chip; circuiti di controllo semplici e regolari; elaborazione pipelining e multielaborazione estese; accoppiamento stretto tra gli algoritmi implementati sui chip e la disposizione dei loro circuiti (Treleaven, 1983). Se verranno realizzate queste condizioni, il collaudo del chip, ovvero la dimostrazione della correttezza della sua struttura, può avvenire su qualsiasi tipo di elemento di elaborazione anziché sull'intero chip. A chip di questo tipo per l'elaborazione specializzata è stato dato il nome di *matrice sistolica* (*systolic array*) (Foster e Kung, 1980). Fra le applicazioni dei processori a matrice sistolica vi sono micropocessori a insieme ridotto di istruzioni, in cui ogni istruzione viene cablata nel chip, chip per la riduzione di grafi o operazioni a flusso, chip progettati specificatamente per operazioni ricorsive e chip che riflettono la struttura ad albero delle operazioni di elaborazione e dei dati.

5.9 Sistemi CAD intelligenti per la progettazione di chip VLSI

Uno dei principali campi di ricerca e sviluppo nell'hardware della quinta generazione sono i sistemi di progettazione assistita dall'elaboratore (CAD, *Computer-Aided Design*) per chip ad altissima integrazione (VLSI). La progettazione di circuiti integrati è già in gran parte automatizzata per mezzo di dispositivi che ne definiscono i vari livelli e permettono di costruire biblioteche di elementi funzionali, controllare i progetti sulla base delle specifiche e simulare il comportamento di un progetto prima di realizzarlo.
Ma ciò che si richiede alla quinta generazione è più di tutto questo, e ancor più complesso: un sistema gerarchico che interagisca con gli ingegneri di progetto a ogni stadio, dalla specifica iniziale alla produzione e alla manutenzione del progetto (Sakamura *et al.*,1982). Lo scopo ultimo è la stesura automatica del progetto, in cui vengono immessi i requisiti logici e di prestazione di un chip, la struttura del chip viene prodotta, simulata e collaudata automaticamente e quindi tramessa direttamente alle apparecchiature di fabbricazione del chip. Il risultato è una fonderia di silicio in cui la progettazione, la messa a punto del prototipo e il ciclo di fabbricazione vengono ridotti da mesi o anni (come accade oggi) a settimane. Un

sistema CAD di questo tipo sarà probabilmente basato su un linguaggio evolut di progettazione del sistema e su interfacce utente standard. Questo è un obiettivo ambizioso che potrà essere realizzato solo quando saranno disponibili i computer della quinta generazione. L'approccio giapponese a questo problema è di natura ciclica, dato che prevede che le versioni intermedie di computer della quinta generazione vengano incorporate nei sistemi CAD che, a loro volta, vengono utilizzati per progettare lo stadio seguente.

5.10 Conclusione

Per molti anni si è parlato di abbandonare la tradizionale struttura del computer proposto da Von Neumann, ma, in pratica, la portata dell'innovazione è stata molto limitata. I diversi livelli di elaborazione parallela necessari alla quinta generazione e una sorta di enfasi posta sull'elaborazione inferenziale anziché sull'elaborazione numerica sembrano dare un certo impulso alla ricerca. Inoltre, per quanto l'obiettivo di realizzare un elaboratore dotato di maggior intelligenza non venga raggiunto, le nuove architetture permetteranno di realizzare degli elaboratori con una potenza che non trova precedenti negli elaboratori tradizionali. È probabile che il passaggio da unità di elaborazione di uso generale ai nuovi aggregati di chip specializzati interesserà tutti i rami della tecnologia delle informazioni. L'aumento del livello di integrazione e l'impiego di sistemi CAD avanzati nella produzione dei microchip troverà applicazioni in tutti i settori della microelettronica. L'aver libero accesso alla nuova generazione di fonderie di silicio comporterà enormi conseguenze industriali, economiche e politiche.

6

Ingegneria del software

Anche se non si dovessero raggiungere gli obiettivi del progetto dei computer della quinta generazione è improbabile che le attuali metodologie di programmazione riescano a sopravvivere a lungo. La scrittura dei programmi secondo i metodi tradizionali è sempre più criticata, e per molti motivi. I programmi sono troppo lunghi: un programma per l'elaborazione di dati commerciali è composto da decine di migliaia di righe in un codice sorgente, il software per l'elaborazione in tempo reale è ancora più lungo e il software della quinta generazione richiederebbe milioni di righe. Via via che i sistemi di elaborazione dati svolgono un numero sempre maggiore di operazioni, l'esigenza di correttezza e affidabilità del software si fa sempre più pressante. I costi del software hanno superato quelli dell'hardware e sono diventati così l'elemento più costoso nella maggior parte dei sistemi di elaborazione. Queste pressioni convergono sulla necessità di sviluppare prodotti software di grandi dimensioni e complessità in grado di soddisfare gli standard di perfetta correttezza e di prestazione in un modo controllato, programmato, preventivato ed economicamente conveniente (Alvey, 1982).
L'avvento della quinta generazione aggiunge tre nuove dimensioni a questo problema. Il software della quinta generazione, servendosi dei metodi dell'intelligenza artificiale per gestire la conoscenza, è basato su concetti e tecniche molto diversi da quelli della tradizionale elaborazione dati. È probabile che l'hardware e il software di molti sistemi elettronici della quinta generazione saranno

progettati insieme, e la distinzione tra i due sarà molto più sfumata di quanto non sia oggi. Per questo e per altri motivi nella maggior parte dei casi non sarà possibile sviluppare il software della quinta generazione sull'unità di elaborazione su cui dovrà poi essere effettivamente eseguito.
Partendo da queste considerazioni, si sta andando verso lo sviluppo di una nuova professione — l'ingegneria del software — che in futuro prenderà il posto della programmazione e probabilmente dell'analisi dei sistemi. Il prodotto finale — i programmi per l'elaboratore — è lo stesso, ma gli strumenti, le tecniche e l'aspetto esteriore sono molto diversi da quelli attualmente in voga.

6.1 La scienza dell'ingegneria del software

Il termine *ingegneria del software* venne proposto per la prima volta ad una conferenza della Nato nel 1968. Esso indica lo sviluppo di programmi per elaboratore in un modo organizzato, soggetto a costi contenuti e a vincoli di tempo e prestazione (Bauer, 1973; Sommerville, 1982). Il termine è stato gradualmente accettato per designare una professione che si trova nel punto d'incontro tra la scienza e l'ingegneria. L'ingegneria del software ha elementi di matematica e logica formali, d'informatica, di economia ed organizzazione e richiede anche una grande esperienza e abilità nella programmazione.
Gli ingegneri del software generalmente lavorano in un ambiente di gruppo. La forza lavoro di un particolare progetto viene ripartita in piccoli gruppi, con una struttura organizzativa che spesso rispecchia la struttura del software stesso (Babich, 1985). L'obiettivo è conciliare le esigenze dell'utente (o acquirente) del sistema con le capacità delle apparecchiature elettroniche nel modo più efficiente possibile. Il lavoro viene programmato in anticipo, preventivato in modo abbastanza dettagliato e sottoposto a un rigoroso controllo di qualità. Per argomenti quali la trasmissione dati sono richiesti alti standard di documentazione interna e compatibilità con gli standard esterni. I problemi che presenta l'organizzazione di un gruppo di ingegneri del software sono, per certi aspetti, senza pari: sono una miscela complessa di problemi tecnici, amministrativi, logistici e personali che devono essere risolti o affrontati molto rapidamente perché il progetto, nel suo insieme, proceda secondo il programma (Brooks, 1975).
I concetti alla base dell'ingegneria del software derivano da una visione a lungo

termine dei cicli di vita del software chiamata *evoluzione del software* (Lehman, 1980, 1982, 1984), che individua i vari stadi durante l'esistenza di un oggetto software — progettazione, sviluppo, attribuzione degli incarichi e quindi cicli di utilizzo, individuazione e correzione degli errori e revisioni di piccola o grossa portata. Adottando questa visione olistica e tentando di attribuire costi ai vari stadi si può capire meglio quali vantaggi si potranno ottenere con le varie tecniche dell'ingegneria del software.
I rimanenti quattro paragrafi di questo capitolo ritornano ai quattro aspetti centrali dell'ingegneria del software: struttura dei programmi, tecniche di progettazione dei programmi, dimostrazione della correttezza dei programmi e ambienti di sviluppo del software.

6.2 Struttura dei programmi

Tutti ormai hanno capito che la chiave per sviluppare software efficiente è progettare molto attentamente la struttura dei programmi. I criteri per avere un programma ben strutturato possono essere riassunti nel seguente modo:

- Una struttura generale chiara basata su moduli, in cui ogni modulo svolge un compito specifico.
- Un'interfaccia chiaramente definita tra i moduli.
- Ogni modulo deve essere una combinazione semplice di costruzioni elementari del linguaggio di programmazione.
- È essenziale raggiungere un alto grado di corrispondenza tra la struttura del programma di un modulo e la struttura dei dati sui quali opera.
- Ogni modulo deve lasciare le strutture di dati su cui opera in uno stato conforme alle proprietà che le definiscono.
- L'attività di qualsiasi modulo particolare non deve causare degli effetti collaterali — non deve alterare alcun dato al di fuori del suo campo di competenza.

Queste sono le regole base della programmazione strutturata, molto più facili da enunciare che da mettere in pratica. Le generazioni successive dei linguaggi di programmazione più avanzati hanno incorporato sempre più i concetti della

strutturazione dei programmi. Nei linguaggi quali l'Algol, il Pascal e l'Ada la tendenza è stata quella di "isolare" sempre più i moduli programma (realizzati come procedure e funzioni) e di fornire canali ben definiti attraverso cui trasmettere reciprocamente i dati. Questa tendenza viene ulteriormente accentuata nella sua realizzazione nel linguaggio di programmazione della quinta generazione Occam, in cui ogni processo è costruito in modo da essere eseguito su un elemento di elaborazione fisicamente distinto.
I requisiti di struttura dei programmi assumono un'altra dimensione nel contesto degli sviluppi del software effettuati dai gruppi di ingegneri del software. La struttura di un programma deve essere abbastanza chiara da permettere che parti diverse possano essere sviluppate indipendentemente. Le ricerche indicano che i problemi dei programmi di grosse dimensioni iniziano e finiscono in un punto semplicissimo — un programma di grandi dimensioni è un programma scritto da più di una persona.
Se si riuscirà a realizzare l'ideale di un vera struttura di programma, si potranno trarre molti vantaggi. Infatti, i moduli potranno essere tenuti in biblioteche per essere riutilizzati; potranno essere messi alla prova separatamente e, qualora fossero scorretti o al di sotto degli standard, potranno essere staccati e sostituiti senza causare effetti collaterali. Tutti i progressi più recenti negli ambienti di sviluppo del software e negli strumenti di generazione dei programmi richiedono una struttura ben precisa del software che si sta sviluppando. Se l'idea di sviluppare hardware e software integrati diventerà realtà, la struttura del software dovrà confrontarsi con un compito ancor più arduo: una stretta corrispondenza con la struttura hardware.

6.3 Progettazione del programma

La progettazione del programma è una metodologia per percorrere il tortuoso e lungo cammino che ha inizio dalla specifica di un programma, generalmente vaga e incompleta e contenente contraddizioni nascoste, e termina con la produzione dell'oggetto in codice macchina, strettamente strutturato, che ha superato tutte le prove cui è stato sottoposto da parte del programmatore e dell'utente. È un procedimento in cui si aggiungono in modo ordinato sempre più dettagli e maggior precisione e, nello stesso modo ordinato, si può ripercorrere lo stesso cammino per

apportare eventuali variazioni qualora risultasse che alcune parti del progetto finiscono in un vicolo cieco. È un procedimento a più cicli di elaborazione in cui ogni ciclo richiede un input da parte dei programmatori, degli utenti e talvolta di quanti si occupano del controllo di qualità. Attualmente vengono utilizzate varie tecniche di progettazione dei programmi. Alcune sono piuttosto informali, e richiedono che la descrizione di un progetto software venga espressa per mezzo di precisi algoritmi. Altri sono molto più formali e si servono di linguaggi di programmazione ben definiti, o di diagrammi per esprimere gli stadi intermedi di un progetto. Sono proprio questi metodi formali a destare l'attenzione dei vari progetti della quinta generazione in quanto sembrano più adatti ad essere sottoposti a prove teoriche che dimostrino la correttezza dei moduli.
Una delle tecniche attuali più semplici e maggiormente diffuse per la progettazione dei programmi è quella del perfezionamento graduale (*stepwise refinement*), nome e metodo forniti da Niklaus Wirth.
Il perfezionamento graduale è una tecnica algoritmica che può essere meglio descritta secondo la sua stessa formulazione:

> Definire i cicli funzionali generali di un programma con un algoritmo succinto e di alto livello.
>
> Ripetere
>
>> Sviluppare ogni passo dell'algoritmo come un algoritmo dettagliato che specifichi i passi della sua implementazione.
>
> Fino a quando il compito è stato specificato in modo abbastanza dettagliato da poter scrivere il codice del programma.

Viene messo in evidenza lo sviluppo discendente, dall'alto verso il basso (*top-down*), in cui i dettagli vengono aggiunti in modo ordinato. I vantaggi di questa tecnica includono la sua flessibilità, l'impiego di una notazione piuttosto informale, e il suo stretto legame con linguaggi di programmazione procedurale come Pascal e Ada. Spesso è possibile utilizzare il testo finale dell'algoritmo quale punto di partenza del programma. Gli svantaggi del perfezionamento graduale sono dovuti alla sua informalità, che lo rende poco adatto ad essere sottoposto a prove formali

o a controlli per dimostrare la correttezza negli stadi intermedi e alla sua impostazione procedurale che non porta naturalmente all'uso di linguaggi di programmazione non procedurali come Prolog.
Un metodo più formale, anch'esso basato sullo sviluppo della struttura di un programma dall'alto verso il basso, è la scomposizione funzionale con tipi di dati astratti. Questo metodo dapprima identifica le proprietà essenziali astratte dei dati gestiti dal programma e specifica l'attività del programma in base alle funzioni operanti su questi tipi di dati astratti. Via via che il progetto evolve, le funzioni vengono definite in base a funzioni più dettagliate e, allo stadio finale, vengono scritte procedure "primitive" per implementare le proprietà astratte dei dati. I vantaggi di questo metodo applicativo sono l'approccio rigoroso alla struttura logica delle funzioni e dei dati, e l'essere adatto a dimostrazioni formali di correttezza a tutti gli stadi. Esso si associa bene con linguaggi di programmazione applicativi quali Hope (e con l'elaboratore a riduzione dei grafi Alice che li utilizza direttamente). Lo svantaggio del metodo del tipo di dati astratti è che si discosta molto dall'aspetto esteriore di un compito di programmazione e dalla struttura hardware della maggior parte dei computer tradizionali. Di conseguenza, è difficile per un profano capire o controllare i cicli di progettazione di un programma; inoltre il software sviluppato secondo questa tecnica può funzionare in modo non sempre efficiente sugli elaboratori attualmente in uso. Comunque, la sua rigorosità e la sua formalità lo rendono un candidato molto promettente per un approccio di ingegneria del software ai compiti di elaborazione della quinta generazione.
Una variazione sul tema dei tipi di dati astratti è il metodo dell'elaboratore di stato, noto col nome di tecnica di Parnas. Questa richiede che si specifichino formalmente lo scopo, le eccezioni ed il valore restituito di ogni funzione e di ogni operazione di un programma. Le operazioni e le funzioni vengono ricavate attraverso un procedimento di graduale perfezionamento partendo da un algoritmo di alto livello. Il vantaggio di questo metodo è che la specifica di ogni modulo è distinta dal procedimento di valutazione del modulo. Di conseguenza, in molti casi è possibile dimostrare che l'algoritmo scelto per un modulo soddisfa la specifica del modulo stesso. Un approccio lungo queste linee è incluso nel linguaggio di programmazione Ada, in cui ogni modulo ha specifiche e un corpo di codice distinti. La specifica descrive l'interfaccia presentata dal modulo al resto del programma, e fa sì che tutti gli accessi al modulo avvengano attraverso l'interfaccia specificata (Barnes, 1982).

6.4 Come si dimostra la correttezza dei programmi

Gli attuali metodi di controllo dei programmi consistono nel sottoporre i moduli programma a una serie di prove manuali e/o automatiche nel maggior numero di condizioni possibili, e nel confrontare i risultati con quelli attesi in base alle specifiche del modulo. Il problema è che, per quanto complete siano, le prove non possono mai simulare tutte le possibili permutazioni dei valori dei dati e degli stati del programma. Pertanto, quasi sempre si parte dal presupposto che ogni programma sia corretto fintanto che questa ipotesi viene confutata da eventuali errori incontrati durante l'elaborazione. Dato che i programmi diventano via via sempre più complessi e vengono utilizzati in situazioni critiche, questo metodo, per quanto esaurientemente possa essere applicato, non sarà più sufficiente. È necessario infatti arrivare ad una dimostrazione logica e convincente della correttezza dei moduli di programma in tutte le particolari condizioni d'uso (Boyer e Moore, 1981).
Le tecniche per dimostrare la correttezza formale dei moduli di programma sono ancora ai primordi, e sono state applicate al software operativo solo pochissime volte. Comunque, costituiscono un punto fondamentale per lo sviluppo di software della quinta generazione attendibile e pertanto sono state oggetto di studio di molti gruppi che si occupano dello sviluppo della quinta generazione. Il problema allo stato attuale delle ricerche è la gamma ristretta di moduli di programma a cui è possibile applicare le dimostrazioni matematiche formali, e il numero limitato di linguaggi di programmazione che si prestano a qualsiasi forma di automazione del processo di dimostrazione.
Una tecnica che può essere applicata in alcuni casi è l'induzione matematica. Questa dapprima stabilisce la validità di una proposizione in casi semplici, come, ad esempio, una struttura di dati vuota; quindi dimostra che se la proposizione è vera in ogni situazione arbitraria, allora essa è vera anche nella "prossima" situazione. ("Prossima" in questo contesto può significare il valore successivo di un intero, o l'aggiunta di un dato a una struttura di dati.) Ad esempio, si consideri la struttura di dati nota come lista, ampiamente usata nei lavori di intelligenza artificiale, che ha l'operatore per aggiungere un dato in testa alla lista. Una lista possiede le seguenti proprietà di definizione minima:

```
lista = lista_vuota
        or cons(dato,lista)
```

```
testa(cons(dato, lista)) = dato

coda(cos(dato, lista)) = lista
```

L'aggiunta di una funzione che concateni due liste può essere realizzata per mezzo del seguente modulo in un linguaggio di programmazione simile al Pascal:

```
function concatena (lista_1,lista_2):lista;

begin
    if lista_1 = lista_vuota
        then concatena:= lista_2
        else concatena:= cos(testa(lista_1),
                             concatena(coda(lista_1),lista_2))
end;
```

Si consideri la proposizione secondo cui la funzione aggiunta, come realizzata sopra, sia associativa:

```
concatena(concatena(lista_1, lista_2),lista_3)=
                             concatena(lista_1,concatena(lista_ 2,lista_3))
```

Il primo stadio di una dimostrazione per induzione è considerare un caso semplice, come, ad esempio, lista_1 = lista_vuota. Utilizzando la definizione della funzione aggiunta data nell'algoritmo per la sua realizzazione, la parte sinistra della proposizione diventa:

```
    concatena(concatena(lista_vuota,lista_2),lista_3)

=   concatena(lista_2, lista_3)
```

La parte destra diventa:

```
    concatena(lista_vuota,concatena(lista_2,lista_3))

=   concatena(lista_2,lista_3)

=   Parte Sinistra
```

Questo stabilisce la proposizione in un caso semplice. Se ora si suppone che la proposizione sia valida per ogni valore arbitrario di lista_1, allora è necessario dimostrare che essa è vera quando un ulteriore dato viene aggiunto a lista_1, dando cons(dato, lista_1). Nell'ultimo caso la parte sinistra della proposizione diventa:

```
     concatena(concatena(cons(dato,lista_1),lista_2),lista_3)

  =  concatena(cons(testa(cons(dato,lista_1))
        concatena(coda(cons(dato,lista_1)), lista_2),lista_3)
```

... dalla definizione di concatena

```
  =  concatena(cons(dato,concatena(lista_1,lista_2)), lista_3)
```

... dalla definizione di testa e coda

```
  =  cons(dato,concatena(concatena(lista_1,lista_2), lista_3))
```

... dalla definizione di concatena e di testa.

La parte destra della proposizione è:

```
     concatena(cons(dato,lista_1),concatena(lista_2,lista_3))

  =  cons(dato, concatena(lista_1,concatena(lista_2,lista_3))
```

... dalla definizione di concatena

```
  =  cos(dato,concatena(concatena(lista_1,lista_2),lista_3))
```

... supponendo che la proposizione sia vera per un valore arbitrario di lista_1.

```
  =  Parte Sinistra.
```

Risulta così che la parte destra della proposizione è uguale alla parte sinistra della

proposizione nel caso di cons(dato, lista_1) se è vera per lista_1. Dato che è si è già dimostrato che è vera quando lista_1 è la lista vuota, allora la proposizione è vera per tutti i valori di lista_1.
L'esempio sopra riportato illustra alcuni dei vantaggi e degli svantaggi del metodo induttivo utilizzato per dimostrare la correttezza dei moduli di programma. Esso implica concetti e tecniche di matematica formale probabilmente sconosciuti a molti programmatori. Inoltre, può essere applicato con buone probabilità di successo su tipi di dati astratti definiti da proprietà semplici. Comunque, per quanto le dimostrazioni possano essere lunghe e noiose, valgono più di qualsiasi prova specifica cui si sottopone un modulo programma con valori di dati reali. Questa tecnica, quando utilizzabile, determina la correttezza degli enunciati di un modulo di programma per tutti i valori possibili dei singoli dati all'interno della loro gamma specificata.
Altre tecniche di verifica dei programmi comprendono generatori di condizioni di verifica automatici, che possono essere applicati a linguaggi come il Fortran, per produrre condizioni formali che ogni modulo, per essere ritenuto corretto, deve soddisfare. Queste condizioni vengono quindi applicate ai sistemi di dimostrazione dei teoremi che stabiliscono se sono vere o false. Un altro metodo consiste nell'utilizzare un linguaggio di specifica diverso (quale il linguaggio Clear) per enunciare le proprietà necessarie a un modulo e per sottoporre la sua specifica a un procedimento di dimostrazione formale. Una terza alternativa è utilizzare metafunzioni (funzioni di funzioni) per controllare la correttezza di un programma funzionale. L'uso dell'induzione matematica nell'esempio sopra riportato è un caso particolare della tecnica delle metafunzioni.
Dimostrare la correttezza dei programmi sequenziali è già cosa difficile; ma il problema si complica ulteriormente quando viene introdotto un elemento parallelo. Il punto è che la logica classica, che sta dietro alla dimostrazione dei programmi, è basata su un ambiente statico: le affermazioni sono comunque o vere o false, e nessun agente esterno può alterare questi valori. Il parallelismo viene esaminato estendendo la logica tradizionale alla logica temporale che ammette l'evoluzione dei sistemi nel tempo (Boyer e Moore, 1981) e di qualche sviluppo recente di sistemi di logica alternativi (Turner, 1984).

6.5 Ambienti di sviluppo del software

Oggi gli strumenti a disposizione degli addetti al software sono una serie di compilatori, concatenatori, dispositivi per la manutenzione e l'indicizzazione delle biblioteche di un modulo, maschere per la progettazione delle videate e dispositivi per la messa a punto quali funzioni di traccia, banchi di prova per moduli individuali e analisi passo per passo. Gli attuali strumenti di sviluppo del software della "quarta generazione" sono varie forme di generatori di programmi, in cui un programma viene impostato in un linguaggio di specifica e il codice viene generato automaticamente. Questi sono l'ideale per alcuni tipi di procedure, quali i pacchetti per l'addestramento a domanda e risposta, ma hanno una scarsa gamma di applicazioni. È comunque ben noto a tutti i gruppi di sviluppo della quinta generazione che saranno necessari strumenti di sviluppo software molto più potenti per realizzare veramente il progetto di un software della quinta generazione che sia economicamente conveniente (Hawley, 1986).
Ciò di cui si ha bisogno è un ambiente di supporto alla programmazione integrato (IPSE, *Integrated Programming Support Environment*), "un insieme compatibile di strumenti basati su una metodologia per tutte le fasi di sviluppo e di funzionamento del sistema, che sia di supporto alle attività sia tecniche che organizzative" (Alvey, 1982). Un IPSE è destinato a fungere da supporto alle unità software durante l'intero ciclo di progettazione, sviluppo, utilizzo e manutenzione.
Il progetto di un IPSE è centrato su un database che comprenda sia tutti i moduli di tutti gli elementi software in via di sviluppo, che le biblioteche dei moduli, le specifiche e le definizioni intermedie dei progetti. Durante la fase di progettazione, sia in caso di un programma nuovo che di modifiche da apportare al software esistente, sono necessari dispositivi per controllare la correttezza e completezza delle specifiche e per facilitare la dimostrazione di correttezza degli enunciati relativi alle specifiche stesse. Via via che vengono specificati i moduli e scritti i codici che li realizzano, sono necessari dispositivi per mettere alla prova ciascun modulo separatamente e agevolare la fase di dimostrazione della effettiva realizzazione di quelle specifiche da parte di quei codici.
È necessaria una matrice dei dispositivi di compilazione, cosicché per ogni applicazione possa essere utilizzato il codice sorgente più adatto e i moduli scritti in linguaggi sorgente diversi possano essere concatenati nello stesso programma. Sono inoltre necessari generatori di programmi per aspetti di routine come il

trasferimento di archivi di dati e l'input/output di testi a video. Quale supporto ai vari elementi di elaborazione per i quali viene sviluppato il software è necessaria una gamma di linguaggi di arrivo. Inoltre, per agevolare la fase di assemblaggio di programmi di grandi dimensioni a partire dai loro elementi costitutivi, è necessario un insieme potente di dispositivi di indicizzazione, riferimento incrociato e concatenamento. Per finire, servono simulatori delle unità di elaborazione obiettivo affinché i codici compilati e concatenati possano essere fatti "girare" in presenza di dispositivi di messa a punto come funzioni di traccia, di elaborazione per passi e via dicendo. In questo modo molti errori di esecuzione possono essere eliminati prima che il software venga passato all'unità hardware finale.
La realizzazione fisica di un IPSE è una rete, locale o geografica, di stazione di lavoro per ingegneri del software, collegate a una o più unità di elaborazione e di gestione di database. Per collegare i vari assemblaggi o prototipi di hardware d'arrivo sono necessarie delle disposizioni flessibili di trasferimento. Tutto questo è un compito molto arduo, soprattutto perché l'hardware d'arrivo avrà, probabilmente, un alto livello di funzionamento in parallelo e la progettazione dell'hardware e del software potrebbe essere un processo simultaneo. È già ovvio che sarà molto costoso sia sviluppare che utilizzare gli ambienti integrati di supporto della programmazione. A meno che non venga messa a punto qualche forma di partizione di tempo, il loro utilizzo sarà limitato a grosse organizzazioni e verranno così praticamente esclusi dall'attività della quinta generazione tutti i programmatori minori.

6.6 Conclusione

Il passaggio, lento ma costante, dalla programmazione all'ingegneria del software sarà probabilmente uno dei maggiori cambiamenti nell'elaborazione dei dati dopo lo sviluppo dei linguaggi più avanzati. Le crescenti richieste di sviluppo di software altamente affidabile a costi accessibili stanno iniziando ad esercitare una certa pressione su individui, società, istituti accademici e nazioni. Se nei prossimi dieci anni le aziende di produzione di sistemi informativi (Alvey, 1982) per hardware e software diventeranno la norma, i programmatori di stampo tradizionale potrebbero fare la stessa fine degli operai addetti all'assemblaggio delle automobili o dei compositori manuali nelle tipografie. Ogni società, istituto o nazione che non si adegui ai nuovi sviluppi verrà relegata nell'informatica di serie B.

7

Linguaggi di programmazione della quinta generazione

L'architettura dei computer della quinta generazione e la gamma di applicazioni cui sono destinati comportano grosse differenze tra i linguaggi di programmazione per il software della quinta generazione e i linguaggi usati oggi. I linguaggi di programmazione tradizionali come il Fortran, il Cobol e il Pascal sono stati sviluppati partendo da architetture a unità di elaborazione sequenziale singola e da applicazioni che richiedevano l'elaborazione di dati. I linguaggi di programmazione della quinta generazione dovranno gestire architetture parallele di alto livello e diversi tipi di unità di elaborazione per la gestione della base di conoscenza, l'elaborazione inferenziale e il supporto delle interfacce intelligenti, e applicazioni basate su sistemi intelligenti basati sulla conoscenza. Essi dovranno essere conformi alla filosofia dell'ingegneria del software, con le sue necessità di costi accessibili, generazione di codici di qualità controllata, dimostrazioni della correttezza dei codici e sviluppo di software in ambienti integrati di supporto alla programmazione. Inoltre, i linguaggi di programmazione della quinta generazione devono avere una solida base nei concetti di intelligenza artificiale che si celano in tutti gli aspetti dei sistemi elettronici della quinta generazione.
La linea generale di sviluppo dei linguaggi di programmazione che tiene conto di queste esigenze è costituita dalla gamma di linguaggi che possono essere definiti non procedurali. I tradizionali linguaggi di programmazione sono procedurali: essi infatti descrivono, in modo molto dettagliato, i passi che un calcolatore deve

compiere per effettuare l'elaborazione richiesta. In altre parole, specificano in che modo eseguire un determinato compito. Sono sequenziali e considerano la memoria dell'elaboratore una specie di lavagna elettronica, su cui l'informazione può essere scritta in qualsiasi momento, e l'informazione esistente può essere ricoperta dai nuovi valori forniti nel corso dell'elaborazione. I linguaggi non procedurali si basano sulla descrizione di ciò che è richiesto da un compito di elaborazione, e della struttura dei dati necessaria a quel compito, ma lasciano al processore del linguaggio i dettagli riguardanti il modo in cui va eseguito il compito. Le differenze tra la descrizione dei dati e le esigenze di un compito di elaborazione non sono sempre chiaramente evidenti e le caratteristiche del compito, secondo la descrizione fornita dal programma, non vengono necessariamente interpretate in sequenza. Molti linguaggi non procedurali si basano sul principio degli assegnamenti non distruttivi: una volta assegnato, il valore di un particolare dato non viene alterato durante l'esecuzione del modulo di programma in cui esso ricorre. Questa proprietà è molto importante quando l'architettura dell'elaboratore non è basata su una memoria statica, ma sul fluire dei dati da una memoria all'altra.
In questo capitolo vengono esaminati due tipi di linguaggi non procedurali: i linguaggi dichiarativi e i linguaggi applicativi (per quanto quest'ultimi sono spesso considerati un sottoinsieme dei primi). Vengono inoltre trattati il linguaggio dell'intelligenza artificiale, Lisp, che ha costituito il punto di partenza di gran parte dell'attività svolta sui linguaggi non procedurali, e nuovi sviluppi nei linguaggi procedurali che potrebbero renderli adatti a certi sviluppi del software della quinta generazione.

7.1 Lisp

Il linguaggio di elaborazione di liste, Lisp, fu creato nel 1959 all'interno del gruppo che, sotto la guida di John McCarthy, si occupava di intelligenza artificiale al MIT (McCarthy,1965; Sammet, 1969). Esso nasce dalle idee di Church sul lambda calcolo (paragrafo 1.2) ed è volto alla gestione di problemi che comportano la manipolazione di simboli e la ricorsione. Gran parte delle ricerche sull'intelligenza artificiale rientra ampiamente in questa categoria: da qui il frequente uso del Lisp per questi scopi (Narayanan e Sharkey, 1985).
L'elemento fondamentale dei dati in Lisp è l'atomo, che può essere un identificatore

o un numero. Mediante parentesi, gli atomi si uniscono a formare liste. Ad esempio, (A B C) è una lista di tre atomi e (A(BC)D) è una lista contenente due atomi e una sottolista (BC). La notazione (A | B) indica una lista con l'atomo A come testa e l'atomo B come coda. Le strutture di dati costruite in questo modo vengono definite espressioni-S.

L'elaborazione in Lisp è basata su funzioni scritte come espressioni-M, tra le quali ve ne sono cinque elementari:

car[(ABC)] = A		... la testa della lista
cdr[(ABC)] = (BC)		... la coda della lista
cons[(A;(BC)] = (ABC)		... la funzione di costruzione
eq[(A;A] = T	(per Vero)	... uguaglianza di atomi
atom[((ABC)] = Nil	(per Falso)	... test per un atomo

(I nomi delle funzioni car e cdr derivano dalle relative istruzioni di linguaggio assemblatore sul primo computer su cui venne implementato il Lisp. Talvolta vengono chiamate rispettivamente testa e coda.)

Le operazioni aritmetiche vengono espresse come funzioni: sum [A;B] dà la somma degli atomi A e B. Le relazioni aritmetiche vengono espresse come funzioni con valori booleani, o predicati: lessp[A;B] è vero se A è minore di B. La costruzione condizionale è

```
[pl—>el; p2—>e2 ... pn—>en]
```

dove p1, p2 ...pn sono predicati ed e1, e2 ...en sono espressioni. La costruzione viene analizzata sequenzialmente e viene valutata la prima espressione corrispondente a un predicato che sia vera. C'è anche una costruzione più tradizionale if...then...else, strutturata come una lista:

```
(if A B C)
```

Se il predicato A è vero allora viene scelto il valore di B, altrimenti viene selezionato il valore di C.

La costruzione define (definisci) crea nuove funzioni, che vengono quindi incorporate nell'ambiente del programma in questione.

Un programma in Lisp viene scritto come una funzione di alto livello che, durante la sua implementazione, chiami funzioni di livello inferiore. Le funzioni possono essere scritte anche come espressioni-S che consentono la loro elaborazione per mezzo di altre funzioni, e forniscono ai programmi Lisp infinite possibilità di automodificarsi. Un esempio di funzione scritta con un'espressione-S è la seguente:

```
(define member(atom_1 list_2)
    (if((consp(list_2))
        (if eq(atom_1 car(list_2))
            T
            (member(atom_1 cdr(list_2))))
        Nil))
```

La funzione sopra definita controlla se atom_1 è un membro di list_2. Se list_2 è non-vuoto (controllo effettuato dalla funzione consp che, nella maggior parte delle versioni del Lisp, è predefinita), allora se atom_1 è uguale alla testa di list_2 la funzione è vera, altrimenti la funzione è chiamata nuovamente a controllare se atom_1 è un membro della coda di list_2. Nel caso in cui list_2 sia vuota, la funzione è falsa, il che è denotato dal valore Nil.
Per molti anni il Lisp è stato il principale linguaggio di programmazione dell'intelligenza artificiale, e quasi sempre i programmi di intelligenza artificiale più importanti sono stati scritti in Lisp. Il Lisp è stato il punto di partenza di molti fra i più recenti linguaggi di programmazione non procedurali, in particolare del Prolog. Per quanto i livelli "superiori" di software di un computer della quinta generazione saranno probabilmente di natura dichiarativa (si veda il paragrafo 7.5), ai livelli inferiori è necessario un linguaggio procedurale che possa operare su strutture di dati di grandi dimensioni e complessità. Il Lisp, o un suo derivato, è un candidato ideale per svolgere questo ruolo. Un'evoluzione possibile del Lisp è verso un linguaggio applicativo (paragrafo 7.7), senza assegnamenti distruttivi, che lo renda compatibile con le architetture a flusso di dati e di riduzione dei grafi.

7.2 Linguaggi procedurali: nuovi sviluppi

Nella loro forma attuale, i linguaggi di programmazione procedurali non trovano

posto nella quinta generazione di computer, ma alcuni progressi compiuti recentemente, soprattutto l'incorporazione del parallelismo e la comparsa del linguaggio Occam, indicano che la linea di sviluppo iniziata con Algol nel 1957 potrebbe continuare verso calcolatori del futuro. Come già spiegato nel paragrafo 5.2, un certo grado di parallelismo "controllato" può essere incorporato nei linguaggi di programmazione procedurali. Questa impostazione venne adottata per la prima volta nel Pascal concorrente, ed è stata adottata nel linguaggio per il tempo reale Ada. L'implementazione più avanzata e completa del parallelismo nei linguaggi procedurali è nell'Occam, in cui ciascun modulo è considerato un processo di comunicazione sequenziale che può funzionare sul proprio elemento di elaborazione possibilmente dedicato. Per fornire alcuni dettagli relativi a questi sviluppi, nei seguenti due paragrafi vengono presentate le caratteristiche dell'Ada e dell'Occam.

7.3 Ada

L'Ada è stato sviluppato tra il 1975 ed il 1980 come progetto finanziato dal Dipartimento della Difesa americano. La versione finale del linguaggio venne scritta da una gruppo guidato da Jean Ichbiah alla CII Honeywell Bull. L'Ada è destinato ad essere il principale linguaggio di sviluppo per sistemi in tempo reale, quali i sistemi di guida dei missili, e a diventare il linguaggio standard di tutti i sistemi di questo tipo del Dipartimento della Difesa americano e forse anche di altri paesi della Nato. Si può immaginare che, con la produzione di nuove versioni, i sistemi elettronici di questo tipo avranno incorporato un grado maggiore di intelligenza e l'Ada potrà così diventare, per mancanza d'altro, un linguaggio di programmazione della quinta generazione.
L'Ada è un grosso linguaggio volto a sviluppare programmi complessi (Barnes, 1982; Young, 1984). Esso è fornito di funzioni ausiliarie per la strutturazione dei programmi e dei dati, tra cui la tipizzazione forte e dispositivi di astrazione dei dati, che fanno sì che le proprietà logiche di una struttura di dati possano essere separate dalla sua implementazione particolare. Il concetto di tipo generico permette di creare moduli di programma indipendenti dai tipi di dati in elaborazione. Ci sono meccanismi per la compilazione separata dei moduli e per la gestione delle biblioteche di moduli come un compito per l'elaborazione in parallelo. Fin dall'inizio l'Ada è stato progettato per essere utilizzato in un ben preciso ambiente

di sviluppo: l'Apse (*Ada Program Support Environment*, Ambiente di supporto dei programmi in Ada).
Per fornire un esempio di un pezzo di programma in Ada, si consideri una procedura che raccolga tre segnali in entrata e quindi li utilizzi per svolgere un certo compito di elaborazione. I tre segnali in entrata provengono da tre fonti indipendenti e possono arrivare in qualsiasi sequenza, in serie o in parallelo.

```
procedure signal_processing is

    task get_signal_1 is
        entry receive (signal_1:out signal);
    end get_signal_1;

    task body get_signal_1 is
        received_signal:signal;

    begin
        received_signal:=signal_channel_1;
        accept receive (signal_1:out signal)do
            signal_1:=received_signal;
    end get_signal_1;
```

... definizioni simili per get_signal_2 e get_signal_3...

```
begin
    get_signal_1.receive(sig_1);
    get_signal_2.receive(sig_2);
    get_signal_3.receive(sig_3);
    process_signal(sig_1,sig_2,sig_3);
end signal_processing;
```

I tre compiti (get_signal_1, get_signal_2 e get_signal_3) sono attivati in parallelo all'inizio della procedura. Ciascuno di essi chiama funzioni particolari (signal_channel_1, signal_channel_2 e signal_channel_3) per ricevere i segnali. Ogni compito quindi giunge all'incontro indicato dalla costruzione *entry...accept*. Se il

compito ha un segnale prima della chiamata all'incontro (*rendez-vous*) nel mezzo della procedura, rimane in attesa all'incontro fino a quando la chiamata viene raggiunta. In modo analogo, se la chiamata raggiunge un incontro prima che il segnale sia pronto, la procedura chiamante rimane in attesa finché il compito chiamato ha emesso il segnale. Quando sono stati ricevuti tutti e tre i segnali viene chiamata la procedura process_signal.

L'Ada è stato ampiamente criticato per le sue dimensioni e per la scarsa eleganza delle sue costruzioni. Per quanto sia progettato in modo da poter essere sottoposto a metodi formali per dimostrare la correttezza dei moduli di programma, rimane ancora molto da fare prima che ciò sia fattibile su vasta scala. La perplessità riguardo all'Ada nasce dal fatto che un errore in un sistema Ada può comportare conseguenze catastrofiche. Ciò nonostante, il Dipartimento della Difesa continua a insistere sull'impiego dell'Ada in tutte le future applicazioni in tempo reale. Attualmente sono in via di sviluppo alcuni ambienti di supporto alla programmazione in Ada, e l'esperienza maturata su questi sarà certamente di aiuto all'attività di ingegneria del software della quinta generazione.

7.4 Occam

Occam è un linguaggio di programmazione volto a fornire un supporto all'elaborazione in parallelo nei sistemi elettronici in cui molte unità di elaborazione funzionano l'una indipendentemente dall'altra, ma interagiscono (Inmos, 1984). Esso è stato sviluppato da Tony Hoare all'Università di Oxford, ed è il linguaggio macchina del transputer Inmos (paragrafo 5.6). Come l'Ada, può essere utilizzato per sistemi in tempo reale, ma è molto più semplice; il suo nome deriva dal filosofo medievale a cui si deve il rasoio di Occam, uno strumento intellettuale affilato utilizzato per tagliare tutti i dettagli superflui in un sistema.

L'Occam è basato sull'idea di processi sequenziali comunicanti (Hoare, 1978). Un processo esegue una sequenza di azioni e può comunicare con altri processi per mezzo di precisi canali. La comunicazione lungo il canale è sincrona: quando sia un processo di input che un processo di output sono pronti a comunicare sullo stesso canale, il dato viene trasferito e i processi continuano indipendentemente. Il concetto generale dei processi comunicanti è illustrato nella figura 7.1. In Occam i tipi di dati possono essere solo numeri interi, caratteri, matrici e stringhe.
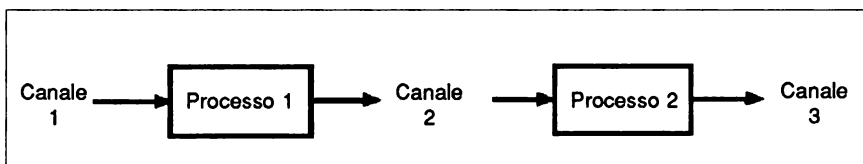
Fig. 7.1 Processi sequenziali comunicanti.

Quale esempio, consideriamo due semplici dispositivi hardware, uno che raccolga i segnali provenienti da tre canali in entrata ed il secondo che accumuli l'output dal primo prima di trasmetterlo al canale di uscita. Si veda la figura 7.2. Un programma in Occam che controlli questa configurazione avrà questo andamento:

```
chan link
par
    while true
        var x:
        alt
            c1 ? x
                link ! x
            c2 ? x
                link ! x
            c3 ? x
                link ! x
    while true
        var x:
        seq
            link ? x
            c4 ! x
```
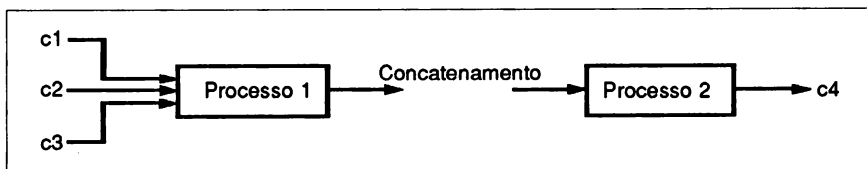
Fig. 7.2 Esempio di configurazione hardware per programmazione in Occam.

Le due costruzioni while true operano in parallelo come cicli senza fine. All'interno della prima costruzione la parola chiave alt seleziona quale dei tre canali è pronto per l'input. Le costruzioni di input (c1 ? x indica l'input del valore di x dal canale c1, ecc.) controllano gli output corrispondenti (link ! x, indica l'output del valore di x sul canale chiamato link). Il secondo processo ripete all'infinito la sequenza di input dal canale link e output verso c4. Si noti che la struttura di un programma viene creata quasi completamente con i rientri: il campo d'azione dei costruttori par, while, alt e seq sono le righe rientrate al di sotto di esse.

La virtù dell'Occam è la sua semplicità. Esso infatti ha meno di trenta parole chiave e solo pochi costruttori. Per quanto ogni processo utilizzi assegnamenti non distruttivi, l'uso di canali di comunicazione tra i processi lo rende del tutto compatibile con le architetture a flusso di dati e di riduzione dei grafi. Occam è stato progettato tenendo conto di questo tipo di architetture e nella prospettiva delle future applicazioni della quinta generazione. Esso è in armonia con la filosofia della quinta generazione secondo cui l'hardware e il software di un sistema di elaborazione dati vanno sviluppati insieme (paragrafo 4.5). Con il transputer Inmos, (paragrafo 5.7) fornisce un componente modulare hardware/software del tipo necessario nella costruzione di sistemi ad alto grado di funzionamento in parallelo. Per la sua semplicità, l'Occam è ben visto quale futuro mezzo di controllo della correttezza dei processi. La sua mancanza di dispositivi per la strutturazione dei dati e la sua stretta vicinanza all'hardware di un sistema di elaborazione dati lasciano pensare che, probabilmente, l'Occam diventerà il linguaggio macchina dei sistemi della quinta genrazione, con applicazioni scritte in un linguaggio più astratto. Questo è il caso per esempio, del computer Alice (paragrafo 5.6).

7.5 Linguaggi dichiarativi

Un linguaggio come l'Occam rappresenta l'idea della programmazione procedurale portata agli estremi delle attuali architetture dei calcolatori elettronici. Ciò nonostate rimane un linguaggio che rientra nella tradizione di "un insieme di istruzioni per controllare le operazioni di un computer". Esso stabilisce i particolari passi necessari per svolgere un determinato compito secondo le capacità di elaborazione elementare di un computer. La comunità dell'intelligenza artificiale si è opposta a questo tipo di approccio per molti anni: molti professionisti esperti

preferiscono infatti descrivere i compiti di elaborazione in base alle proprietà logiche intrinseche dell'informazione (o conoscenza) e alle trasformazioni che debbano essere eseguite su queste informazioni (Kowalski, 1982). Questa linea di ricerca ha portato ai linguaggi di programmazione procedurali, dei quali il Prolog costituisce l'esempio più significativo.
La necessità dei linguaggi di programmazione dichiarativi ha comportato notevoli complicazioni. Ha infatti determinato uno spostamento del centro d'attenzione dal calcolatore al compito di elaborazione. Ha inoltre fatto sorgere un'ampia gamma di nuove applicazioni che prima non erano considerate accessibili per l'elaborazione elettronica. Fra queste vi sono alcuni tipi di sistemi esperti, le nuove impostazioni delle basi di dati (Kowalski, 1984), l'uso della logica quale metodo di avvicinamento alla programmazione nel settore dell'istruzione (Kowalski, 1982; Ennals, 1983), e il problema generale dell'elaborazione delle inferenze per mezzo di grandi basi di conoscenza che possono essere incomplete, imprecise e contraddittorie. La metodologia dichiarativa, che usa il Prolog come punto di partenza, è stata adottata in Giappone dal gruppo di sviluppo della quinta generazione quale base della propria attività di ricerca. Il linguaggio centrale della quinta generazione, KL0, usato dal gruppo dell'Icot, è basato sul Prolog (Aiso, 1982).

7.6 Prolog

Il Prolog è stato sviluppato da Alain Colmerauer e dai suoi colleghi all'Università di Marsiglia nel 1972. Si basa sull'idea generale che l'elaborazione è una deduzione controllata e un algoritmo è una combinazione di logica e controllo (Campbell, 1984; De Saram, 1985). L'aspetto logico del Prolog è l'uso di costruzioni del tipo:

```
A se B e C
```

note col nome di clausole di Horn, in cui ogni elemento è un predicato del tipo:

```
necessario-per(iodio, sviluppo-tiroide)
```

come spiegato nel paragrafo 2.3. I predicati che contengono costanti, come nell'esempio sopra riportato, appartengono alla base di informazioni usata dal

programma, mentre quelle con variabili, come:

```
causa-deformità(x,y)
```

apprtengono al compito di elaborazione. In questo modo, sia gli aspetti dell'elaborazione sia quelli della strutturazione dei dati della programmazione tradizionale vengono espressi con un'unica notazione basata sulla logica. I dati nei predicati possono essere dati singoli e liste con le stesse regole di contruzione utilizzate nel Lisp (paragrafo 7.1). Ad esempio, l'appartenenza a una lista viene definita per mezzo delle clausole:

```
membro-di(x,(x | z))
membro-di(x,(y | z)) se membro-di(x,z)
```

(In altre parole, un dato x è membro di una lista se è la testa della lista o membro della coda della lista.)
Un programma in Prolog consiste in una successione di clausole di Horn contenenti predicati del tipo sopra riportato, e viene attivato da domande del tipo:

```
quale(x,causa-deformità(x,crescita-stentata)) .
```

Un esempio di programma Prolog che utilizza un semplice database di elementi chimici è il seguente:

```
elementi-di(acqua,(idrogeno ossigeno))
elementi-di(ammoniaca,(azoto idrogeno))
elementi-di(metano,(carbonio idrogeno))
elementi-di(anidride-carbonica, (carbonio ossigeno))
elementi-di(alcool,(carbonio idrogeno ossigeno))
membro-di(x,(x | z))
membro-di(x,(y | z)) se membro-di(x,z)
componente-di(x,y) se elementi-di(y,z) e membro-di(x,z)
```

L'ultima clausola viene interpretata come "x è un componente di y se y ha una lista di elementi z ed x è un membro della lista z".

Una domanda quale

```
quale(y,componente-di(carbonio,y))
```

che significa "Quale composto ha il carbonio tra i componenti?" produce i risultati:

```
metano
anidride-carbonica
alcool
nessuna (altra) risposta.
```

L'elaborazione viene svolta per mezzo di un procedimento di dimostrazione di teoremi per risoluzione (Robinson,1965), in base al quale le clausole di Horn vengono trasformate in un formato interno equivalente, i predicati "cancellati" fino a quando si ottiene un risultato o sorge una contraddizione, nel qual caso la domanda non ha risposta. Nonostante la natura non procedurale del Prolog, il procedimento di risoluzione è sequenziale dato che esso prende le clausole di Horn secondo l'ordine in cui compaiono nel programma. Questo determina un certo numero di problemi, soprattutto legati alla ricorsione "stretta" e alla rappresentazione di informazioni negative. Ad esempio, nella maggior parte delle implementazioni del Prolog, clausole del tipo:

```
connesso-a(x,z) se connesso-a(x,y) e connesso-a(y,z)
```

e

```
connesso-a(x,y) se connesso-a(y,x)
```

faranno sì che l'interprete cada in un ciclo senza fine.
La negazione nel Prolog viene interpretata come "fallimento" o "insuccesso" del tentativo di stabilire che un predicato è vero. Così

```
non(componente-di(x,y))
```

è vero se tutti i tentativi di stabilire

componente-di(x,y)

falliscono. Secondo le parole di Robert Kowalski, "Purtroppo l'implementazione della negazione per mezzo dell'insuccesso rendono il successo o l'insuccesso di una condizione negativa pericolosamente sensibili al contesto in cui viene eseguita" (Kowalski, 1982).
Problemi di questo tipo con la forma originale del Prolog sono oggetto di ricerche in numerosi centri, dove si considerano attentamente gli sviluppi conseguenti a questi studi dato che saranno proprio essi a determinare se i linguaggi di programmazione dichiarativi basati sulla logica continueranno a essere fondamentali nella quinta generazione. Tra gli sviluppi recenti ci sono il Parlog, un'implementazione parallela del Prolog, e vari tentativi di sintetizzare elementi del Prolog e del Lisp, o il Prolog e un linguaggio procedurale. Il piano di sviluppo per il linguaggio di programmazione dell'Icot prevede un linguaggio sequenziale a base logica, KL0, seguito da una versione parallela, KL1, e da successive versioni adatte all'evoluzione dell'hardware dei sistemi della quinta generazione.

7.7 Linguaggi applicativi

I linguaggi applicativi sono una classe di linguaggi di programmazione che precedono i computer della quinta generazione, ma che possono rientrare nel filone della nuova generazione. Hanno avuto origine dall'evoluzione delle idee sulla strutturazione dei programmi e dal bisogno di procedure formali per la progettazione e la verifica dei programmi. Appartengono alla classe dei linguaggi di programmazione funzionali la cui origine risiede nel lambda calcolo di Church (paragrafo 1.2) e sono in contrasto con i linguaggi tradizionali imperativi basati su algoritmi che determinano come calcolare un risultato.
I linguaggi applicativi si basano sugli effettivi risultati dell'elaborazione, senza prestare troppa attenzione ai singoli passi procedurali. Un programma in un linguaggio applicativo viene considerato un insieme di oggetti di dati, anziché un insieme di "ricette" per ottenere i valori dei dati.
A prima vista, le caratteristiche fondamentali di un linguaggio di programmazione applicativo possono sembrare restrittive. I programmi consistono interamente di funzioni e non ci sono procedure. Gli assegnamenti sono non distruttivi: quando una

variabile ha assunto un valore, lo mantiene per tutto il resto del modulo. Questo significa che non possono coesistere due assegnamenti che abbiano lo stesso nome di variabile nella loro parte sinistra. Gli enunciati in un linguaggio applicativo non devono essere interpretati in sequenza: sono tutti validi simultaneamente. Una conseguenza è che in un linguaggio applicativo non ci sono costruzioni iterative. Grazie a queste proprietà i linguaggi applicativi sono adatti allo sviluppo dei computer della quinta generazione per più motivi. Gli assegnamenti non distruttivi, unitamente alle proprietà non sequenziali degli enunciati, significano che essi sono l'ideale per architetture hardware a flusso di dati, a riduzione dei grafi o in generale per l'elaborazione in parallelo. L'essere limitati dalla struttura funzionale fa sì che i programmi possono essere sottoposti a dimostrazioni di correttezza con maggiore facilità. Inoltre, l'interpretazione asincrona degli enunciati rende effettivamente possibile utilizzare macchine specializzate con elementi di elaborazione "tagliati su misura" per determinati tipi di operazioni. Alcuni linguaggi applicativi si basano sulle liste come strutture di dati, caratteristica che permette di collocarli nela corrente principale dei lavori sull'intelligenza artificiale.

7.8 Hope

Un esempio di linguaggio di programmazione applicativo è l'Hope, sviluppato all'Università di Edimburgo e adottato dal gruppo di ricerca dell'Imperial College per il computer Alice (paragrafo 5.5). Un programma in Hope è una funzione che chiama altre funzioni per completare la propria definizione. Ogni funzione descrive un oggetto di dati, che può essere un intero, una stringa di caratteri, una lista o una struttura più complessa. Una funzione può essere definita per un tipo di dati generico e il tipo specifico le viene passato come un parametro. Questo significa che, ad esempio, la stessa funzione ordinerà un insieme di stringhe di caratteri o un insieme di numeri. In Hope tutte le variabili sono vicine alla dichiarazione della funzione nella quale ricorrono; questo significa che i programmi non subiscono effetti collaterali come quando un modulo inavvertitamente cambia il valore di una variabile in un altro modulo. I programmi in Hope non vengono interpretati sequenzialmente e godono della proprietà matematica della trasparenza referenziale (Bailey, 1985, 1986). Questo facilita la costruzione e la dimostrazioni di asserzioni sulla correttezza dei programmi e permette ai programmi di essere

trasformati automaticamente in forme equivalenti che siano più efficienti nell'utilizzare le risorse hardware del calcolatore su cui devono girare.
Una funzione in Hope che inserisca un elemento al suo posto giusto in una lista ordinata viene scritto nel seguente modo:

```
dec insert: alpha#lista(alpha) → lista(alpha)
— insert (i,nil)        ← i::nil;
— insert (i,h::t)       ← if i > h
                              then i::(h::t)
                              else h::insert(i,t);
```

La prima riga dichiara una funzione chiamata insert, che prende come argomenti un singolo dato del tipo non specificato alpha e una lista dello stesso tipo, e dà come risultato un'altra lista. La seconda riga definisce il valore della funzione, dato un elemento i e una lista vuota, come l'elemento i seguito da una lista vuota (la notazione :: corrisponde al | del Lisp). La terza riga definisce il valore della funzione dato un elemento i e una lista con testa h e una coda t in due casi: se i precede h allora la funzione produce la lista con testa i e coda h :: t; altrimenti la testa rimane h e la funzione viene chiamata ricorsivamente per inserire i nella coda t.
Ogniqualvolta viene chiamata questa funzione, per mezzo di un procedimento di confronto di forme si determina quale delle funzioni viene usata (entrambe possono essere controllate in parallelo). Anche le definizioni alternative che corrispondono alle costruzioni then ed else possono essere valutate in parallelo e il valore richiesto viene selezionato quando è noto il risultato del confronto. La funzione è polimorfa, in quanto opera su elementi di dati di tipo non specificato (con la limitazione che la relazione di precedenza valga per il tipo di dati).

7.9 Conclusione

La caratteristica fondamentale dei linguaggi di programmazione della quinta generazione delineati in questo capitolo è che sono molto diversi dai loro immediati predecessori. In particolare essi sono non procedurali, non sequenziali e non numerici e richiederanno capacità di programmazione diverse da quelle attuali, cosa che influirà notevolmente sulla comunità dei programmatori di ogni paese che

aspiri a far parte, nei prossimi dieci anni, dell'industria mondiale dell'informatica. Attualmente non è chiaro se, come è stato ampiamente predetto, il Prolog o uno dei suoi linguaggi derivati diventerà il linguaggio di base della quinta generazione. In questo capitolo abbiamo visto che vi sono parecchi candidati a questo ruolo e che parecchi linguaggi potrebbero essere utilizzati per i vari livelli di software che controllano l'insieme dei processori che compongono un sistema della quinta generazione.

8

Sistemi intelligenti basati sulla conoscenza

I sistemi intelligenti basati sulla conoscenza (*Intelligent Knowledge-Based Systems*, IKBS), termine coniato in occasione dell'Alvey Report (Alvey, 1982), sono gli elementi centrali dei calcolatori elettronici della quinta generazione. Essi sono stati descritti come i sistemi "più vicini all'intelligenza artificiale applicata" (Taylor, 1983) e corrispondono, grosso modo, ai "sistemi applicativi di base" del programma Icot (figura 3.1: Moto-Oka, 1982). I sistemi intelligenti basati sulla conoscenza realizzano l'unione dei vari elementi hardware e software dei computer della quinta generazione per raggiungere la meta verso la quale è diretta tutta l'attività di ricerca e sviluppo: l'uso dell'inferenza applicato alla conoscenza per eseguire un compito. L'eventuale successo o fallimento dei progetti di sviluppo della quinta generazione dipenderanno, in ultima istanza, dalla loro capacità di portare alla produzione a costi accessibili di utili sistemi intelligenti basati sulla conoscenza.

Gli IKBS operano proprio in quelle aree "grige" in cui finora i computer hanno avuto solo un uso limitato. Entrano nel complesso campo dei processi decisionali per collaborare con esperti quali medici, geologi, economisti e responsabili di quasi tutti i settori e per svolgere funzioni finora impossibili, come prendere decisioni tattiche durante il volo di un missile. Essi sono pensati per operare su patrimoni di informazioni e di conoscenze molto vasti e complessi, e per trarre inferenze basate su una combinazione di regole categoriche, ipotesi di lavoro e conclusioni tratte da

operazioni precedenti. Gli esseri umani sono molto abili nel progettare calcoli, ma non nell'eseguirli: gli elaboratori elettronici sono i più recenti, tra i tanti dispositivi in grado di fare calcoli al posto delle persone. Con i sistemi esperti basati sulla conoscenza si cercherà di operare lo stesso trasferimento di mansioni su un campo più vasto: essi infatti implementeranno i piani di elaborazione di conoscenze generali di persone o gruppi di persone.

Per riuscire in questo obiettivo un IKBS deve avere all'interno almeno le seguenti capacità: classificazione, formulazione di concetti, sintesi, astrazione, selezione, filtraggio del ricupero dei dati, ragionamento, pianificazione, costruzione di modelli, memorizzazione e formulazione e utilizzo di regole euristiche ("buon senso").

L'interazione tra un IKBS e il mondo esterno probabilmente avverrà per mezzo di suoni, immagini, attraverso il tatto e soprattutto attraverso la comunicazione in un linguaggio naturale. Sarà pertanto necessario che questi sistemi abbiano la capacità di interagire con l'esterno attraverso l'analisi e la produzione del linguaggio, la percezione e generazione di immagini, la sensazione e la manipolazione degli oggetti fisici (Alvey, 1982). Questi aspetti degli IKBS sono trattati nel capitolo 9.

Il rapporto Alvey identifica un certo numero di possibili applicazioni dei sistemi intelligenti basati sulla conoscenza: diagnosi scientifiche, interrogazioni di basi di dati, insegnamento di abilità elementari, recupero di documenti sulla base del contenuto, programmazione delle attività negli uffici, coordinamento dei robot, combinazione delle immagini, interpretazione dei messaggi, consulenze legali, stesura di programmi, produzione di manuali tecnici, controllo dei processi industriali e imballaggio nei magazzini. Inoltre sono previste applicazioni militari, tra cui sistemi "intelligenti" di missili guidati, sistemi di pianificazione strategica e tattica, reti di comunicazione intelligenti e possibile utilizzo nei sistemi di difesa strategica antimissilistici. Alcuni di questi rientrano nella categoria dei sistemi esperti (paragrafo 10.6); altri sono applicazioni più generali dei computer della quinta generazione.

Non vanno sottovalutate le difficoltà di implementazione dei sistemi intelligenti basati sulla conoscenza. Per il momento mancano i fondamenti teorici sull'argomento. Non esiste un'unità di misura universale per l'intelligenza e neppure operazioni primitive che operino sugli elementi della conoscenza come fa l'aritmetica elementare sui numeri. Citando le parole di Donald Michie, abbiamo bisogno di una "teoria della conoscenza solida e ben quantificata, che attualmente è pressoché inesistente" (Michie, 1982).

Questo capitolo tratta i metodi e le tecniche che sono stati utilizzati per realizzare sistemi intelligenti basati sulla conoscenza, e si sofferma su alcuni dei problemi incontrati.

8.1 Rappresentazione della conoscenza

La chiave per riuscire ad elaborare la conoscenza è la rappresentazione della conoscenza, ossia la capacità di memorizzare in un sistema elettronico patrimoni di grandissime dimensioni di informazioni altamente strutturate in modo da mantenere il "significato" delle informazioni e da poterle elaborare in tutti i modi citati nel paragrafo precedente. La rappresentazione della conoscenza è anche una delle principali sfide che dovranno superare quanti si occupano di sviluppo della quinta generazione. Non è necessariamente affrontabile con tecniche di "forza bruta" che talvolta hanno successo se applicate a problemi di hardware, né seguendo l'impostazione minimalista, altamente disciplinata, che tanto spesso ha fornito brillanti soluzioni ai problemi del software tradizionale e delle basi di dati. Come discusso nei paragrafi 2.2 e 2.3, la conoscenza è complessa, vaga, spesso incompleta e può contenere contraddizioni. La maggior parte della conoscenza è una miscela di informazioni e di vari livelli di istruzioni che controllano l'uso delle informazioni.
Un punto principale del dibattito sulle tecniche di rappresentazione della conoscenza è l'aspetto del controllo: le informazioni di base e gli elementi di controllo devono essere contenuti in un'unica struttura di rappresentazione o in due separate?
Un ulteriore punto di dibattito è costituito dal problema dell'apprendimento. Molti esperti di intelligenza artificiale sono convinti che nessun sistema che non abbia incorporata la capacità di apprendere possa essere descritto come intelligente. In genere ci sono due approcci all'apprendimento — l'approccio deduttivo, in cui la conoscenza specifica viene dedotta da regole generali, e l'approccio induttivo in cui regole generali vengono desunte da esempi specifici. Il problema consiste nel decidere quale metodo, o quale combinazione di metodi, è preferibile in ogni applicazione degli IKBS, e come costruire i meccanismi per implementarli.
Ci sono due impostazioni generali per quanto riguarda la rappresentazione della conoscenza (Steels e Campbell, 1985), definite, rispettivamente, dichiarativista e proceduralista. L'impostazione dichiarativista consiste nel rappresentare l'informazione in maniera neutrale, indipendentemente dal suo utilizzo. Il controllo

viene realizzato ad un livello superiore per mezzo di strategie generali di elaborazione della conoscenza che vengono applicate uniformemente all'informazione. Il linguaggio Prolog è un'implementazione di questo approccio: forma una base di conoscenza partendo da associazioni tra elementi di informazioni e regole di inferenza operanti su queste associazioni e comprende un meccanismo generale di dimostrazione dei teoremi per verificare la corretta applicazione delle regole alle associazioni. Una variazione sul tema dichiarativista include elementi di controllo nella base di conoscneza come entità distinte. L'impostazione proceduralista oscura deliberatamente la distinzione tra informazione e controllo e crea strutture di rappresentazione che comprendono entrambi gli aspetti. Quando la base di conoscenza viene elaborata, il contesto di un particolare processo determina se l'informazione viene presa per quello che è o utilizzata per effettuare controlli.
In questo capitolo vengono presentate tre tecniche specifiche di rappresentazione della conoscenza: reti semantiche, casellari e sistemi di produzione. La quarta fra le tecnica più diffuse, l'uso di una logica di primo ordine, è stata trattata nei paragrafi 2.3, 7.5 e 7.6. Vedremo tre esempi di elaborazione della conoscenza, per avere un'idea dell'attività che queste tecniche comportano e dei problemi da superare: sono la ricerca in una base di conoscenza, il ragionamento basato sull'evidenza e l'apprendimento procedurale.

8.2 Reti semantiche

Le *reti semantiche* (*semantic networks*) considerano la conoscenza come un insieme di associazioni tra concetti (Alty e Coombs, 1984). I concetti vengono rappresentati come i nodi di una rete e le associazioni come gli archi che uniscono i nodi. Diversamente da un programma in Prolog, che elenca ogni associazione separatamente, una rete semantica può avere qualsiasi grado di complessità, con molti archi che si riferiscono allo stesso nodo.
La figura 8.1 illustra una rete semantica che rappresenta le seguenti associazioni:

Giovanni è uno studente che gioca a calcio e a cui piace la musica classica.

Elena è un'impiegata a cui piacciono il calcio e la musica classica.

Giuseppe è un impiegato che gioca a calcio e a cui non piace la musica classica.
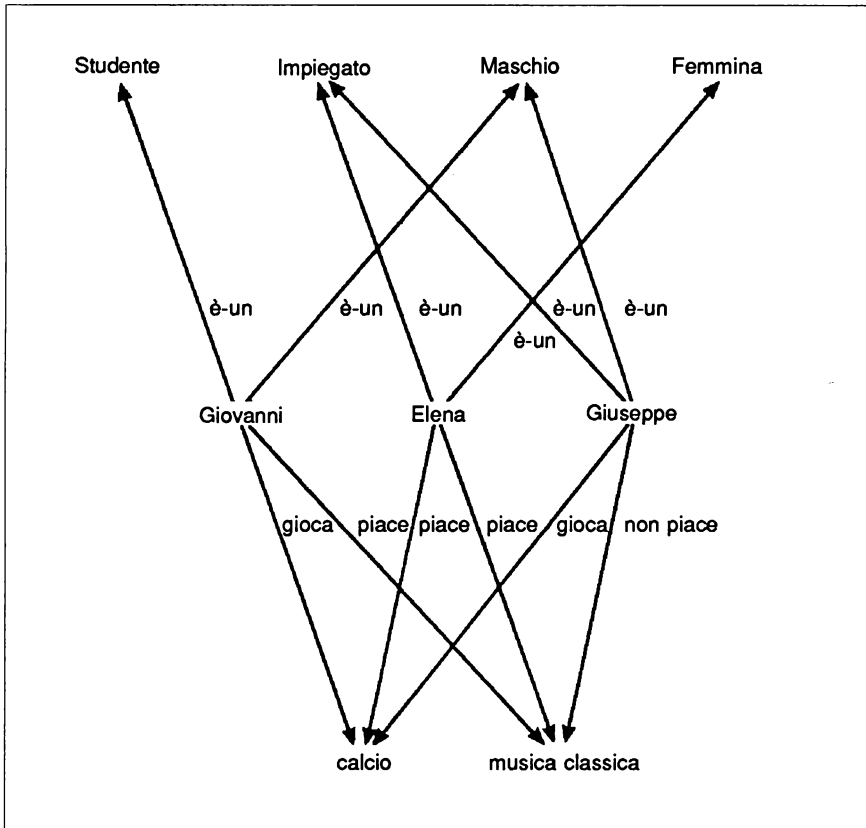
Fig. 8.1 Esempio di rete semantica.

Si noti che tutti i nodi della rete sono nomi e le associazioni sono verbi. I nomi comprendono individui particolari (chiamati elementi) e classi generali (chiamate tipi), con la convenzione di racchiudere tra parentesi i gli elementi per distinguerli dai tipi. I verbi comprendono l'appartenenza a un insieme (è un) che permette di applicare tutte le operazioni della teoria degli insiemi alle reti semantiche. Quello sopra riportato è un esempio di una rete semantica statica in cui le associazioni non mutano nel tempo. Reti semantiche dinamiche possono essere costruite attorno ai verbi come nodi. La figura 8.2 illustra in che modo può essere rappresentato il seguente record di occupazione di una stanza d'albergo:
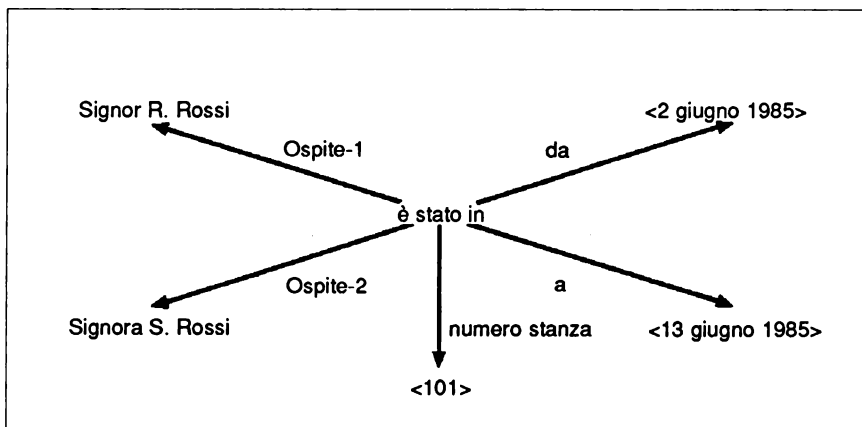
Fig. 8.2 Rete semantica dinamica per l'esempio dell'occupazione della stanza.

> Il signore e la signora Rossi hanno occupato la stanza 101 dal 2 giugno 1985 al 13 giugno 1985.

Le reti semantiche possono essere rappresentate da strutture generali, chiamate *prototipi*, in cui tutti i nodi sono tipi anziché elementi. Il prototipo per la figura 8.2 è illustrato nella figura 8.3.
Il vantaggio delle reti semantiche è la loro flessibilità: infatti possono essere utilizzate per informazioni "pure" in un contesto dichiarativo o includere archi e nodi di controllo in regime proceduralista. Inoltre, si adattano bene ai programmi scritti in Lisp o in linguaggi simili. La loro struttura può essere adattata in modo che le informazioni richieste più frequentemente siano raggiungibili attraverso un percorso di ricerca più breve, mentre quelle richieste più raramente vengono tenute in posizioni più recondite. Sono state utilizzare per analizzare la struttura delle frasi, le relazioni tra sintomi clinici, la relazione spaziale tra oggetti e i sintomi e cause di avaria nei dispositivi meccanici. Lo svantaggio delle reti semantiche è che la loro potenziale complessità va contro la necessità di avere un numero ridotto di principi fondamentali che regolino l'elaborazione di una base di conoscenza. Date le loro vaste dimensioni e la conseguente necessità di aggiornamenti localmente non si prestano facilmente ad essere elaborate da un computer a flusso di dati o a riduzione dei grafi.
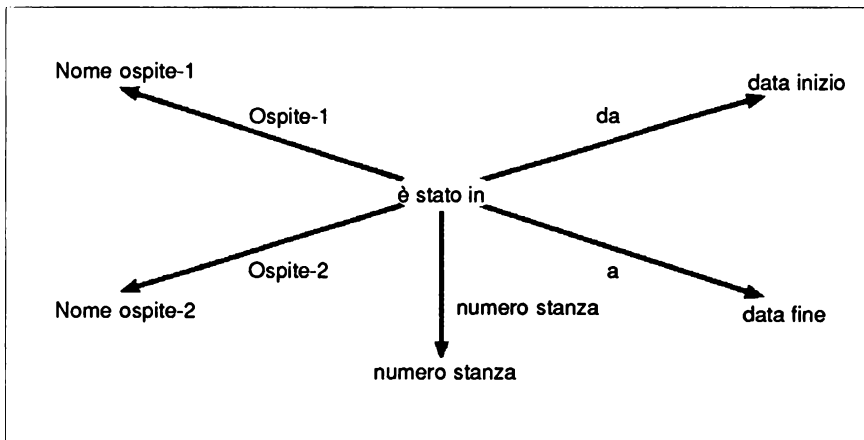
Fig. 8.3 Prototipo di rete semantica per la figura 8.2.

8.3 Casellari

Il concetto di prototipo utilizzato con le reti semantiche porta al metodo di rappresentazione della conoscenza per mezzo di *casellari* o *frames* (Alty e Coombs, 1984). Un casellario è una costruzione che viene ripetuta tante volte quante è necessario per concatenare determinati elementi di dati. Un casellario ha un nome ed un insieme di caselle, ciscuna con un nome. Ogni casella può essere riempita con un singolo dato, con un insieme di dati (in questo contesto chiamato *unità*) o un riferimento a un altro casellario.

Un casellario generale per l'esempio del paragrafo precedente sull'occupazione di una stanza d'albergo è il seguente:

Nome:	SOGGIORNO_STANZA	
	Ospite:	NOME_OSPITE
	Numero_stanza:	Intervallo (da 1 a 199)
	Data_iniziale:	Unità(Giorno,Mese,Anno)
	Data_finale:	Unità(Giorno,Mese,Anno)

La prima casella si riferisce a un altro casellario, illustrato sotto. La seconda casella viene riempita con una specificazione dell'intervallo di valori possibili, la terza e la

quarta casella con le descrizioni degli insiemi di dati che devono contenere.

Nome:	NOME_OSPITE	
	Nome_1:	Unità(Titolo,Iniziale,Cognome)
	Nome_2:	Unità(Titolo,Iniziale,Cognome)

Riempiti con i dati per l'esempio specifico sopra riportato, questi schemi diventano:

Nome:	Codice_stanza_8506_09	
	Ospite:	Ospite_8506_09
	Numero_stanza:	101
	Data_iniziale:	2 giugno 1985
	Data_finale:	13 giugno 1985

Nome:	Ospite_8506_09	
	Nome_1:	Signor P. Rossi
	Nome_2:	Signora R. Rossi

Ai casellari vengono attribuite specifiche identità e vengono inseriti i valori dei singoli dati.
I casellari possono essere riempiti anche con chiamate a procedure di elaborazione e i valori dei dati possono venir passati a e da queste procedure. L'esempio sopra potrebbe essere esteso fino a includere una chiamata a una procedura che calcola il costo del soggiorno, dato il numero della stanza e le date iniziali e finali.
I casellari generalmente vengono utilizzati in un contesto di elaborazione della conoscenza proceduralista, dato che possono contenere sia dati che informazioni di controllo. Sono strutturati in modo più formale rispetto alle reti semantiche, e pertanto sono più adatti ad architetture a flusso di dati o a riduzione dei grafi. I casellari vengono solitamente programmati in Lisp, ma sono adatti anche all'elaborazione con i linguaggi funzionali o applicativi. Il loro limite è dato dalla struttura fissa, che li rende difficili da utilizzare nell'implementazione dell'aspetto dell'apprendimento degli IKBS.

8.4 Sistemi di riduzione

Come trattato nel paragrafo 2.3, un sistema di rappresentazione della conoscenza richiede tre livelli di memoria dei dati: un insieme di associazioni tra gli elementi, che costituisce la base di conoscenza, un insieme di regole per trarre le inferenze derivanti dalle associazioni, e un sistema di controllo per applicare le regole in maniera razionale. Un'implementazione specifica di questa tecnica per mezzo della logica dei predicati di prim'ordine espressa in Prolog è già stata descritta in precedenza (paragrafi 7.5 e 7.6). Una realizzazione più generale di questa impostazione è quella per mezzo di regole di produzione, espresse come enunciati del tipo:

se <condizione> allora <azione> .

In generale, queste regole possono essere poste in una gerarchia in cui l'elemento inferiore opera direttamente sulla base di dati, e livelli superiori di regole operano su livelli inferiori di regole. Possono essere utilizzati diversi tipi di sistemi di controllo, a seconda del tipo di base di conoscenza.
Ad esempio, un sistema di prediagnosi medica interattiva potrebbe iniziare con regole di questo tipo:

MR1 Se domanda_specifica viene chiesta e risposta è sì allora aggiungi contenuto_domanda a base dati.

MR2 Se sintomo_variabile descritto come alto o basso allora misura variabile e aggiungi lettura a base dati

R1 Se identità_paziente non determinata allora determina identità_paziente e leggi riassunto_cartella_medica dal disco nella base dati

R2 Se sintomo_principale sconosciuto allora chiedi sintomo_principale e aggiungi risposta a base dati

R3 Se temperatura > 39 allora chiedi se ha_capogiri.

R4 ...

Le prime due regole sono definite meta-regole MR1 e MR2 dato che vengono applicate ad altre regole e non a oggetti di dati. Le altre risiedono in un livello inferiore e agiscono su informazioni fornite dal paziente o che lo riguardano. Un semplice sistema di controllo per un insieme di regole di produzione di questo tipo potrebbe essere: esaminare le regole in successione a partire da quella di livello più elevato e rendere attiva la prima per la quale la condizione sia vera. Quando la regola è stata applicata, e la base dati aggiornata, l'insieme di regole viene nuovamente esaminato partendo dall'alto.
Applicando queste regole a un paziente che abbia temperatura alta e soffra di capogiri, il primo passo rende attiva la regola R1 per stabilire l'identità del paziente. Quando la cartella medica del paziente è stata letta dal disco, il passo successivo delle regole rende attiva la regola R2, la cui risposta è "temperatura alta". Questo attiva la meta-regola MR2 al passo successivo e la temperatura del paziente viene debitamente misurata. Se è superiore a 39, viene attivata la successiva regola R3, e il paziente risponde "si"' alla domanda sui capogiri. Questo attiva la meta-regola MR1 la quale fa sì che il contenuto della domanda — ha_capogiri — venga aggiunto alla base di dati. E così procede la diagnosi.
Un sistema di produzione di questo tipo sta, in qualche modo, tra il concetto di base di conoscenza proceduralista e quello dichiarativista. C'è una strategia generale di applicazione di queste regole che è indipendente da esse, ma le regole sono organizzate gerarchicamente e quindi tengono conto della strategia di applicazione. Questo rende la tecnica molto flessibile, ma rende molto difficoltoso qualsiasi metodo formale che ne dimostri la correttezza. Per quanto la strategia di applicazione richieda che le regole vengano considerate in successione, una valutazione parallela delle condizioni e la scelta della regola di grado più alto per cui la condizione è vera sarebbe ugualmente efficace. Quest'ultima si presta all'elaborazione in parallelo o a qualche tipo di valutazione per gruppi, qualora siano disponibili un numero di elementi d'elaborazione insufficiente perché l'intero insieme di regole possa essere esaminato subito.

8.5 Elaborazione della conoscenza

Come già accennato all'inizio del capitolo, le operazioni di elaborazione della conoscenza comprendono classificazione, formulazione di concetti, sintesi, astra-

zione, filtraggio del reperimento di dati, ragionamento, programmazione, costruzione di modelli, memorizzazione, formazione e uso di regole euristiche e apprendimento. La maggior parte di queste operazioni sono molto diverse da quelle dell'elaborazione di dati tradizionale, ordinamento, selezione, memorizzazione, recupero di informazioni ed esecuzione di calcoli. Si possono individuare due ampie categorie di operazioni d'elaborazione della conoscenza. Il primo gruppo concerne la formulazione e la manipolazione di regole, che possono seguire una fra due direzioni: quella deduttiva, in cui nuove conclusioni vengono tratte partendo da principi stabiliti, e l'induttiva, in cui nuovi principi generali vengono suggeriti da prove specifiche. In molte applicazioni è comunque probabile che vengano seguite entrambe le direzioni di inferenza. Il secondo gruppo riguarda la ricerca di configurazioni corrispondenti (o quasi corrispondenti) o la creazione e il controllo di eventuali alternative. In pratica, lo spazio di ricerca e il numero di alternative da generare o da controllare è molto alto.

Nell'elaborazione della conoscenza ci sono ancora parecchi problemi da superare prima di poter costruire sistemi intelligenti basati sulla conoscenza, tra i quali gestire informazioni incerte, incomplete e contradditorie e evitare che un'esplosione combinatoria subentri durante l'elaborazione, in particolare durante le operazioni di ricerca. Il primo problema viene studiato per mezzo dell'applicazione della logica "sfumata" basata su insiemi che non hanno confini ben definiti e per mezzo dell'inclusione di probabilità o fattori ponderali nelle regole di inferenza. (Si veda il paragrafo 2.3.)

Il problema delle esplosioni combinatorie tormenta i ricercatori di intelligenza artificiale da molti anni. È stato uno dei principali fattori negativi identificati nel Lighthill Report (Lighthill, 1972) che ha posto fine a molta attività sull'intelligenza artificiale in Gran Bretagna durante gli anni settanta. Un'esplosione combinatoria ha luogo quando il numero di possibilità da considerare durante l'operazione di elaborazione della conoscenza è di gran lunga superiore a quello che il computer può gestire. Talvolta, infatti, non si considera che tutti i computer, compresi quelli che appartengono alla quinta generazione, sono macchine limitate — capaci di immagazzinare una quantità pur sempre limitata di informazioni e conoscenza e di svolgere un numero limitato di operazioni in un dato tempo. Se un'operazione viene intrapresa con un metodo semplicistico come procedere per tentativi con tutte le possibilità, il numero di prove da effettuare, per simulare tutte le possibili situazioni pratiche, ben presto va oltre le capacità anche del più grosso elaboratore elettronico

esistente. Ad esempio, in un sistema di riconoscimento del parlato con un dizionario di diecimila parole, una ricerca non intelligente del significato di una frase di tre parole dovrebbe prendere in considerazione un miliardo di possibilità. Il rapporto Lighthill ha ben sottolineato che il numero di possibilità da considerare in situazioni limitate, come una partita a scacchi, è a malapena gestibile dal più potente degli elaboratori elettronici di oggi, ma nelle applicazioni del mondo reale questo numero sarebbe veramente troppo alto.
Ora il problema dell'esplosione combinatoria sembra meno scoraggiante per due ragioni. Da un lato, le capacità e la velocità di elaborazione dei computer sono aumentate di quasi due ordini di grandezza rispetto al decennio scorso, e sembra possano aumentare di altrettanto entro i prossimi dieci anni. Dall'altro lato, lo studio delle tecniche di elaborazione della conoscenza si sta concentrando molto sui processi di ricerca intelligenti in grado di determinare già in uno stadio iniziale se una linea particolare di ricerca riuscirà a produrre un risultato o no.

8.6 Ricerca in una base di conoscenza

Come discusso nel paragrafo precedente, la ricerca in una base di conoscenza e la creazione e la prova di un alto numero di alternative (che corrisponde quasi allo stesso procedimento) sono operazioni fondamentali di elaborazione della conoscenza. Il campo in cui si svolge la ricerca viene chiamato spazio degli stati della ricerca. Lo spazio degli stati può essere il contenuto di una base di conoscenza, o un insieme di possibili alternative che possono essere generate. Ad esempio, in un'applicazione di gioco lo spazio degli stati è costituito da tutte le possibili mosse del gioco. La struttura più comune di uno spazio degli stati è un albero, che si espande dalle radici che rappresentano lo stato iniziale (o corrente) del sistema e si ramifica per ogni possibilità ad ogni stadio successivo. Gli stati finali del sistema rappresentano gli obiettivi verso cui sono dirette le varie azioni. Un esempio di spazio degli stati in una struttura ad albero di questo tipo è illustrato nella figura 8.4 che rappresenta le configurazioni possibili di un dato sistema di microcomputer.
A seconda delle circostanze ci sono due direzioni possibili in cui si può effettuare una ricerca su un albero. La *concatenazione in avanti* (*forward chaining*) implica il movimento dallo stato corrente alla radice dell'albero verso uno stato obiettivo che deve soddisfare alcune condizioni.
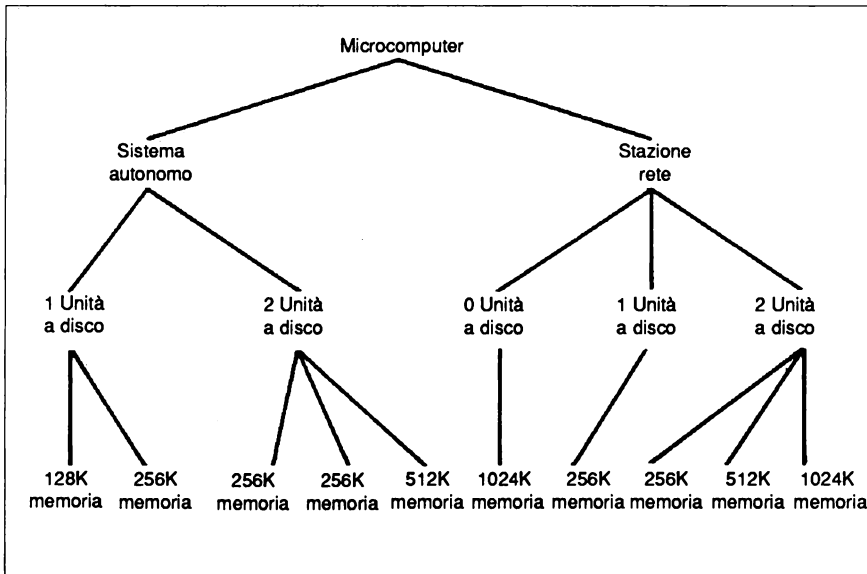
Fig. 8.4 Uno spazio degli stati come struttura ad albero.

La *concatenazione all'indietro* (*backward chaining*) inizia allo stato obiettivo e determina quale percorso deve essere scelto ad ogni nodo per riuscire a raggiungere questo obiettivo. In un'applicazione di gioco ci si serve della concatenazione in avanti per studiare le mosse alternative che iniziano allo stato corrente del gioco, mentre si ricorre alla concatenazione all'indietro per determinare quale successione di mosse porterà a un determinato stato obiettivo del gioco.
Qualunque sia la direzione della ricerca, due sono le strategie utilizzabili. La strategia della priorità alla profondità considera tutte le possibili conseguenze di ogni mossa a qualsiasi stato prima di considerare una mossa alternativa. La strategia della priorità all'ampiezza considera tutte le possibili alternative a ogni stato prima di considerare le conseguenze di ciascuna. Le due strategie sono illustrate nella figura 8.5. Gran parte degli studi attuali puntano alla realizzazione di strategie di controllo sempre più intelligenti per la ricerca, in modo da scongiurare il pericolo delle esplosioni combinatorie. Prima di passare all'analisi dei nodi successivi, viene calcolato in qualche modo il valore o beneficio (o rischio) di ogni nodo intermedio dello spazio degli stati. Se si riuscirà a ridurre le dimensioni dello spazio da studiare

si potrà dire con certezza che il tempo di elaborazione e la fatica richieste da queste valutazioni intermedie non saranno stati spesi male. La maggior parte dei metodi attuali sono una combinazione delle due forme di ricerca con priorità alla profondità e all'ampiezza.
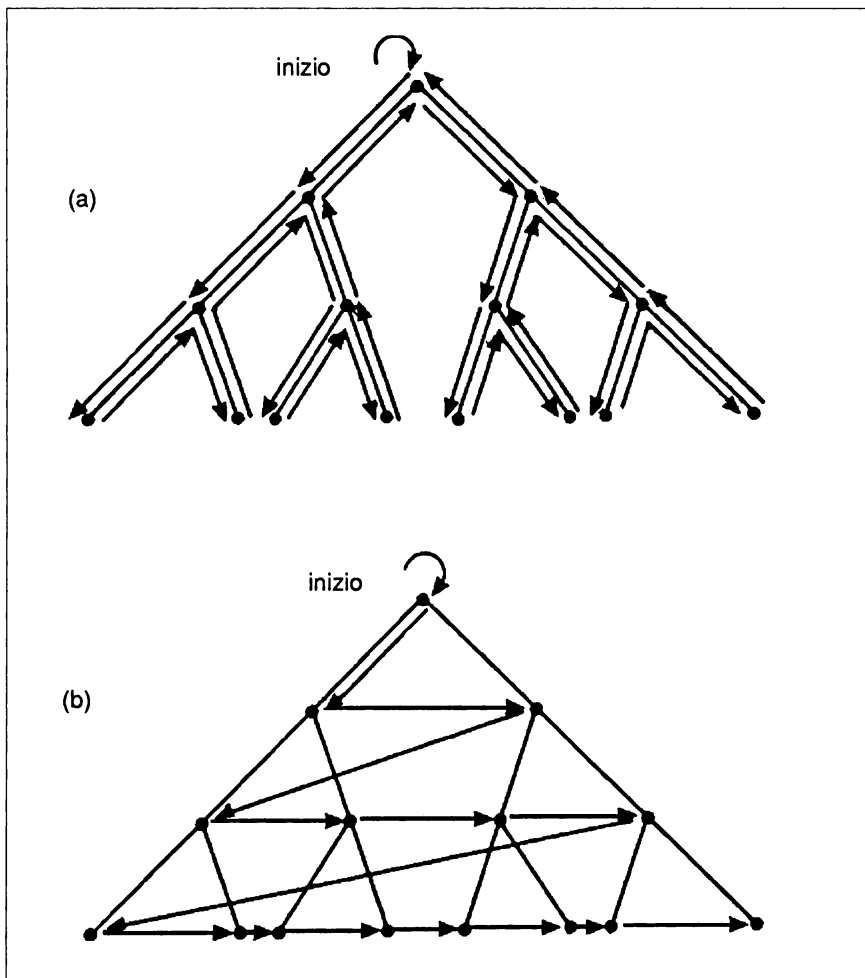
Fig. 8.5 Ricerca con priorità alla profondità (a)
e priorità all'ampiezza (b) in uno spazio degli stati.

8.7 Ragionamento sulla base di evidenza

Un esempio del tipo di elaborazione della conoscenza che verrà intrapreso dai sistemi di elaborazione dati della quinta generazione è il trarre conclusioni da dati di fatto, evidenze. Un complesso di prove evidenti, come quello fornito nel corso di un processo o un'indagine su un'incidente o sullo sviluppo di una proposta è spesso troppo vasto e può essere contradditorio, incompleto, condizionante o deliberatamente fuorviante. Inoltre, l'evidenza viene presentata in forme diverse, come trascrizioni di analisi incrociate, tavole di figure con proiezioni speculative, fotografie, mappe, diagrammi e risultati scientifici e forensi.
Il problema che i ricercatori devono affrontare è quello di assimilare un complesso molto vasto di conoscenze di questo tipo, attribuire un peso relativo agli elementi, risolvere le contraddizioni e trarre conclusioni, o per lo meno arrivare a qualche punto d'accordo sulla situazione descritta dall'evidenza. L'approccio tradizionale per l'elaborazione di dati incerti come quelli di questo tipo è costituito dai modelli di probabilità bayesani, in cui la probabilità delle conclusioni tratte dall'elemento probativo può essere calcolata assegnando probabilità agli elementi probativi originali. Un metodo alternativo (Lowrance e Garvey, 1982), attualmente oggetto di studi, è sviluppare una teoria matematica dell'evidenza e utilizzarla per arrivare a opinioni unanimi partendo da complessi di evidenze e trarre conclusioni da queste opinioni. Questo richiede un meccanismo che permetta di risalire al significato partendo da brani in linguaggio naturale e quindi di elaborare gli enunciati dotati di significato in maniera sistematica, per poter risolvere eventuali contraddizioni e trarre conclusioni. Questo approccio è ancora a livello primordiale, ma costituisce un valido esempio della direzione in cui si stanno evolvendo i sistemi intelligenti basati sulla conoscenza.

8.8 Apprendimento procedurale

Il concetto di apprendimento procedurale (Langley, 1985) è frutto dell'osservazione del modo in cui la gente impara. Quando una persona si trova, per la prima volta, di fronte a un particolare tipo di problema, per affrontarlo possiede solo metodi deboli e di carattere generale, come il procedere per tentativi ed errori e il metodo euristico. Secondo le strategie di ricerca sopra discusse, la persona non

possiede alcun metodo per la valutazione dei nodi intermedi di una ricerca, e deve pertanto tentare moltissime possibilità prima di giungere a una soluzione. Via via che la persona acquista esperienza e conoscenze specifiche, vengono sviluppati strumenti analitici più potenti che permettono di restringere il campo di ricerca sempre più rapidamente. Ad esempio, un medico con anni di esperienza può generalmente arrivare a una diagnosi rivolgendo al paziente solo poche domande o sottoponendolo a pochi esami.

L'apprendimento procedurale è un tentativo di trasferire ai sistemi di elaborazione dati questo processo di perfezionamento delle tecniche di ricerca partendo dall'esperienza. Un sistema intelligente basato sulla conoscenza che usi questa tecnica ha bisogno di numerosi dispositivi interagenti, fra cui la capacità di generare delle alternative in una data situazione e di valutare i risultati. Esiste un sistema di attribuzione causale, in grado di determinare quali situazioni o quali attività sono responsabili di un miglioramento o di un peggioramento del processo di soluzione, e una capacità di modificare le strategie di risoluzione dei problemi col beneficio del giudizio retrospettivo. Utilizzando queste tecniche, un sistema di apprendimento procedurale è capace di risolvere i problemi fin dall'inizio, partendo con solo "forza bruta e ignoranza", ma riflettendo sulle sue azioni durante ogni tentativo, e incorporando via via le sue esperienze nelle sue strategie operative. Man mano che vengono risolti casi particolari di un certo problema, le strategie diventano più efficienti e quindi aumentano le dimensioni dei compiti che si possono intraprendere.

Generalmente la conoscenza, in un sistema di apprendimento procedurale, viene espressa sotto forma di regole di produzione. Il sistema viene creato con regole molto deboli e generiche, ma poi, con l'uso, vengono ricavati insiemi di regole più specifiche ed efficaci. Le regole vengono postulate e quindi conservate o scartate, a seconda dei loro effetti sulle prestazioni del sistema. Le regole sono organizzate gerarchicamente e pertanto includono un elemento di controllo del modo in cui esse vengono applicate. Le auto-valutazioni effettuate determinano la validità delle regole specifiche come pure delle meta-regole che governano la loro applicazione. Ad esempio (secondo Langley, 1985), dovendo risolvere un'equazione algebrica del tipo

$$3x - 4 = 17$$

un primo insieme di regole tratte da un sistema di apprendimento procedurale è il

seguente:

Se funzione_1 ha un argomento num_1
e funzione_2 è l'inverso di funzione_1

allora applica funzione_2 ad entrambe le parti dell'equazione
con num_1 quale argomento.

In questa formulazione, num_1 rappresenta un numero e non un'incognita in un'equazione. Applicata all'equazione sopra, la regola nota che il primo membro include la funzione $f(y) \rightarrow y - 4$, e pertanto applica la funzione inversa $f(z) \rightarrow z + 4$ ad entrambi i membri. Comunque, potrebbe ugualmente aver notato che il primo membro include la funzione $f(w) \rightarrow 3w$ e potrebbe applicare il suo inverso, $f(v) \rightarrow v/3$, che non condurrebbe a una soluzione al primo passo. Dopo ulteriori tentativi ed errori, e numerosi insiemi di regole di produzione in qualche modo difettose, si arriva alla formulazione di un insieme di regole valide in tutti i casi:

Se funzione_1 è al livello esterno dell'equazione
e funzione_1 ha per argomento num_1
e funzione_2 è l'inverso di funzione_1

allora applica funzione_2 ad entrambi i lati dell'equazione
con num_1 quale argomento.

Quando il sistema è arrivato alla formulazione di questo insieme di regole che viene applicato a un'equazione ripetutamente, fino a quando si giunge a una soluzione, ottiene la soluzione corretta per tutte le equazioni dello stesso tipo al primo tentativo.

8.9 Conclusione

I sistemi intelligenti basati sulla conoscenza costituiscono una delle sfide maggiori con cui si devono confrontare i gruppi di sviluppo della quinta generazione. Per quanto sia possibile realizzare implementazioni di IKBS dimostrativi agli stadi

intermedi, solo quando saranno disponibili sistemi avanzati di hardware e software sarà possibile produrre versioni totalmente operative. Il programma Alvey (Alvey, 1982) è stato messo a punto per tenerne conto. Esso infatti prevede la realizzazione di un certo numero di IKBS dimostrativi grandi e piccoli da utilizzare quali studi di fattibilità e mira a destare l'interesse verso i tipi di prodotti che sarà possibile realizzare per mezzo dei sistemi intelligenti basati sulla conoscenza. Allo stesso tempo sta cercando di creare un'infrastruttura di esperti di IKBS in Gran Bretagna e di promuovere ricerche di base, che sono necessarie in gran quantità prima di raggiungere lo stadio di piena realizzazione dei sistemi intelligenti basati sulla conoscenza. Il metodo Icot per gli IKBS (Furukawa, Nakajima *et al.*, 1982) è imperniato sulla programmazione logica quale base della rappresentazione della conoscenza, dei sistemi di elaborazione e dei sistemi di meta-inferenze che li controlleranno. Viene proposto un cammino di sviluppo ciclico con sviluppi di nuove versioni degli IKBS in corrispondenza dei progressi nell'hardware e nel software. Una delle prime applicazioni di ogni realizzazione degli IKBS è coadiuvare lo sviluppo della versione successiva dei propri sistemi di supporto hardware e software.

9

Interfacce utente intelligenti

Il significativo aumento dell'intelligenza dei computer della quinta generazione non è destinato a rimanere nascosto nel profondo dei loro meccanismi di elaborazione delle inferenze e della base di conoscenza. Uno degli obiettivi principali dei programmi di sviluppo della quinta generazione è incrementare il livello di interazione tra l'elaboratore ed il suo utente fino a raggiungere quasi il grado di intelligenza insito nella comunicazione interpersonale. Per far questo è necessario molto di più del concetto piuttosto superficiale di "amichevole con l'utente" che è stato l'obiettivo massimo (raramente raggiunto) dei computer tradizionali. Gli elaboratori della quinta generazione richiedono una "compatibilità cognitiva" (Michie, 1982) con il loro utente.
C'è sempre stata una differenza enorme tra il modo in cui i computer elaborano l'informazione e quello in cui gli esseri umani pensano. I computer della quinta generazione restringeranno un po' questo abisso, ma esso rimarrà ugualmente una linea divisoria significativa. Lo stato attuale delle interfacce tra un sistema di elaborazione e il suo utente può essere descritto come in figura 9.1. Il confine tra il computer ed il suo utente è molto vicino al modo in cui opera il computer, e di conseguenza richiede notevoli sforzi da parte dell'utente per riuscire a comunicare con l'elaboratore. Detto in altri termini, le attuali interfacce utente richiedono che l'utente faccia del suo meglio per riuscire a pensare come gli elaboratori. Le interfacce utente intelligenti, come illustrato nella figura 9.2, stanno cercando di

costruire dei "ponti basati sulla conoscenza" tra "partner che sono intrinsecamente incompatibili" (Michie, 1982). Esse fanno sì che per l'utente sia molto più facile operare con il computer, ma per far questo i computer devono fare il loro meglio per pensare come il loro utente, il che non è poco.

Uno dei principali motivi per cui si dà tanta importanza alle interfacce utente intelligenti è che i computer della quinta generazione sono destinati a essere utilizzati da moltissime persone che, per la maggior parte, non saranno specialisti di informatica. I computer della quinta generazione sono infatti destinati a essere utilizzati negli uffici, nei negozi e nelle fabbriche, come nei laboratori di scienziati, ingegneri, medici ed altri professionisti specializzati. In Giappone i requisiti sociali dei computer della nuova generazione sono enormi, e costituiscono parte integrante della politica sociale generale del paese per i prossimi decenni.
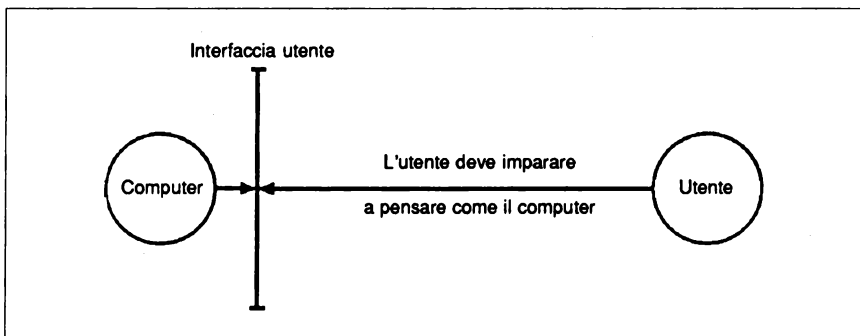
Fig. 9.1 Interfaccia utente tradizionale.
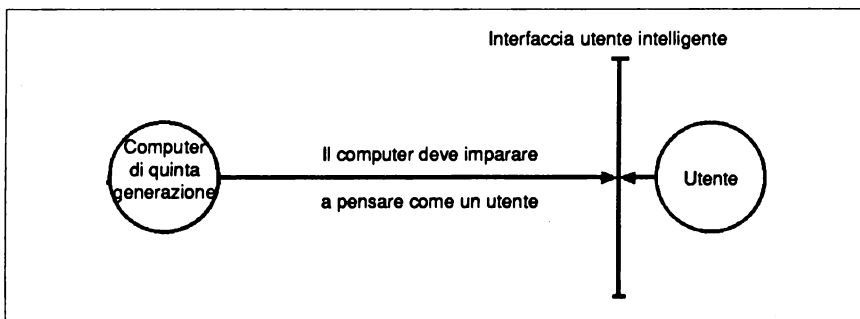
Fig. 9.2 Interfaccia utente intelligente.

9.1 I computer e i loro simboli

Le persone comunicano, tra l'altro, anche per mezzo di simboli visivi e verbali che assumono varie sfumature di significato a seconda del contesto dell'interazione. La relazione tra i concetti e la loro realizzazione in simboli comunicabili costituisce un aspetto fondamentale della psicologia umana ed è oggetto di studio da molti anni. Le interfacce degli attuali elaboratori sono basate su un sottoinsieme molto ridotto di questi simboli e cioè su numeri o rappresentazioni di numeri, diagrammi tecnici, singole parole in un contesto di controllo e sequenze più estese di parole vengono riconosciute da un utente come linguaggio naturale, ma non così dal computer, per i quali sono solo dei dati. Si sta incominciando ad utilizzare semplici icone quali simboli di controllo. Attualmente, l'output di un computer viene solitamente presentato all'utente unicamente per mezzo di simboli stampati o emessi a video, e l'utente fornisce l'input al computer unicamente premendo pulsanti corrispondenti ad un certo insieme di simboli, per esempio agendo sui tasti di una tastiera. Le interfacce utente della quinta generazione stanno per allargare notevolmente l'ambito di questa interazione simbolica. Per ogni particolare applicazione, l'interfaccia utente deve essere parte integrante del sistema nel suo insieme, e deve utilizzare la forma più adatta per comunicare con l'utente. Ad esempio, i robot intelligenti saranno probabilmente dotati di un certo grado di discriminazione visiva, e di senso del tatto. Entrambi questi sensi riguarderanno la percezione, da parte del robot, dello spazio tridimensionale in cui opera. Si prevedono, comunque, grandi progressi nella comunicazione tra il computer e l'utente mediante il linguaggio sia scritto che parlato. Per operare sugli aspetti umani dell'interazione uomo-computer, psicologi e linguisti stanno lavorando con gli informatici al progetto della nuova generazione di interfacce.
Per fornire la profondità dell'intelligenza e gli stretti legami con i sistemi intelligenti basati sulla conoscenza, necessari alle interfacce utente della quinta generazione, è indispensabile un notevole supporto hardware e software. I vari livelli dei meccanismi di supporto delle interfacce intelligenti nei modelli dei computer giapponesi della quinta generazione sono illustrati nella figura 4.1. Nella maggior parte dei casi questi corrisponderanno a grandi, veri e propri sistemi di elaborazione della conoscenza semiautonomi.
Oltre ai consueti chip ad altissima integrazione necessari ai meccanismi di supporto, per la "superficie" dell'interfaccia serviranno molti nuovi dispositivi hard-

ware: video piatti ad alta risoluzione, cineprese e proiettori, sensori infrarossi, raffinati sintetizzatori di suono ecc.

9.2 Interfacce della quinta generazione

Ci si può fare un'idea della tendenza che porta alle interfacce utente intelligenti esaminando la gamma attualmente disponibile di interfacce della quarta generazione che stanno incominciando ad entrare nell'uso. Esse puntano a migliorare la comunicazione concettuale tra l'utente ed il computer a livello visivo e a ridurre al minimo il controllo da tastiera, necessario per operare un programma. Permettono di utilizzare i computer in maniera molto più flessibile rispetto al passato e di ridurre il predominio delle tastiere tradizionali quale mezzo di input e di controllo. L'insieme di questi meccanismi viene chiamato Wimps, sigla che sta per finestre (*window*), icone (*icons*), mouse e puntatori (*pointers*).
Le finestre (porzioni dello schermo utilizzate per compiti diversi da quelli del resto dello schermo) danno l'impressione che il computer stia eseguendo più di un compito alla volta. Molti dei compiti svolti dagli utenti non rientrano esattamente nelle categorie del software per computer attualmente disponibile. Ad esempio, un utente può aver bisogno di redigere annotazioni basate sulle cifre in un foglio elettronico: la porzione necessaria del foglio elettronico può essere visualizzata su una finestra mentre l'utente scrive le note per mezzo dell'elaboratore di testi.
Le icone costituiscono un tentativo di allontanarsi dal predominio delle parole scritte (o visualizzate) nel controllo dei programmi. Al posto di una parola o di una frase contorta in gergo informatico, viene presentata all'utente una serie di semplici icone che mostrano le opzioni di controllo. Ad esempio, invece della frase "cancella l'archivio" o "*delete file*", l'utente ha davanti le icone di un archivio a cartelletta e di un cestino di rifiuti. L'uso di icone rende un programma indipendente da qualsiasi tipo di linguaggio, permette che certi programmi siano utilizzati da analfabeti e aiuta a migliorare la compatibilità cognitiva tra l'utente e il computer.
Il mouse è un dispositivo di controllo complementare alle icone. Esso permette di usare senza una tastiera programmi che utilizzano immagini o configurazioni di controllo simili. L'utente muove il mouse su una superficie piana causando così il movimento di un puntatore o di un segnale luminoso in direzione corrispondente sullo schermo. Premendo un pulsante sul mouse viene attivata la scelta indicata dal

puntatore. La combinazione di mouse, del puntatore e delle icone fornisce un meccanismo di controllo molto più vicino al modo in cui le persone pensano rispetto al "dialogo" piuttosto scarno e carico di espressioni gergali, che con i suoi meccanismi di controllo limitati e fissi è divenuto ormai obsoleto.
Queste interfacce della quarta generazione costituiscono un grande passo avanti e sono il primo sintomo del fatto che, nell'interazione tra uomo e computer, ci si sta occupando seriamente del lato umano. Per le interfacce della quinta generazione, con le loro radici nel linguaggio naturale e nell'intelligenza artificiale, sarà necessario compiere un passo avanti ancora più grande.

9.3 Sintesi del parlato e riconoscimento della voce

L'uso del parlato nell'interazione tra computer e utente è una delle principali mete specificate nei programmi per i computer della quinta generazione. Pur trattandosi di un aspetto completamente diverso, nella pratica la sintesi del parlato è strettamente associata al riconoscimento del linguaggio naturale, oggetto del prossimo paragrafo. Se in questi settori si otterranno dei buoni risultati, al punto da rendere possibile una vasta gamma di applicazioni, ne deriveranno notevoli vantaggi. Comunque, la difficoltà del compito non va sottovalutata. La lingua inglese contiene molti gruppi di due o tre parole che vengono pronunciate allo stesso modo ma hanno un significato diverso. In altre lingue, soprattutto in cinese, il problema è ancor più complesso. Per illustrare questo aspetto, un eminente linguista cinese ha scritto un testo in cui ogni parola viene pronunciata quasi allo stesso modo. Una volta superata la barriera del riconoscimento delle parole in base al suono, rimarrà il problema dell'interpretazione delle parole nel contesto.
I principi della rappresentazione di un segnale acustico in forma digitale sono ben noti. Il meccanismo comunemente accettato è campionare l'onda sonora a intervalli regolari, misurare la sua ampiezza e trasformare la sequenza dei numeri ottenuti in un altro insieme di numeri che indicano l'intensità delle varie frequenze che compongono il suono. Viene utilizzata la tecnica di trasformazione di Fourier. Una nota relativamente pura, quale il suono di un flauto, è per lo più un'unica frequenza con poche frequenze superiori o armoniche. Una parola pronunciata, comunque, è un insieme complesso di frequenze, ognuna delle quali persiste per un certo periodo di tempo. Le variazioni di altezza, dell'accento e dell'enfasi rendono difficoltoso

stabilire un'unità digitale accettata da tutti per le parole che vengono pronunciate. Se la codificazione o la decodificazione dei segnali verbali deve svolgersi in tempo reale, il computer dovrà avere una notevole potenza di elaborazione e una grossa memoria. Ad esempio, per un suono di alta qualità della durata di un secondo sono necessari più di cento kilobyte di dati.

La sintesi del parlato — la produzione da parte del computer di parole o passi più lunghi in un linguaggio naturale — è un compito più semplice del suo inverso, il riconoscimento della voce. I meccanismi fondamentali per la sintesi del parlato partendo da un testo memorizzato sotto forma di caratteri sono già quasi completamente stabiliti. Gruppi di caratteri vengono interpretati come allofoni — gli ingredienti base del suono delle parole. Vengono recuperate le rappresentazioni in forma digitale dei suoni di questi allofoni e sistemate in sequenza. Esse vengono quindi fatte passare attraverso un processo di raffinamento in modo da condurre le sillabe una nell'altra in modo naturale prima di decodificarle in una rappresentazione analogica del parlato, pronte per essere trasmesse. Un sistema di sintesi del parlato a tre livelli messo a punto dalla Digital Equipment Corporation (Bruckert *et al.*, 1983) segue questo approccio. Attualmente l'attività in questo settore è volta soprattutto ad aumentare il vocabolario piuttosto limitato dei sistemi di sintesi del parlato, e a colmare alcune lacune quali la mancanza di un ritmo naturale e del fraseggio. Si stanno introducendo anche variazioni di altezza, accento e intonazione.

Il riconoscimento della voce costituisce un problema che porterà i computer della quinta generazione all'estremo delle possibilità. Prendere un flusso di parole emesse una di seguito all'altra, decodificarlo e interpretarlo, è un processo che interessa tutti gli aspetti dell'hardware e del software. Alcuni progetti per il riconoscimento della voce sono già iniziati, ma non hanno ancora ottenuto grossi risultati; tra questi vi sono il sistema di comprensione del parlato Darpa Hearsay II, (Erman *et al.*, 1980) e il sistema Logica Logos. Tutto quanto si è riusciti a conseguire è il riconoscimento di un vocabolario ristretto di parole e frasi pronunciate da un parlante noto, la cui voce è stata "appresa" dal sistema di elaborazione dati.

Sta diventando piuttosto chiaro che, per riuscire a riconoscere la voce, è necessario un approccio di tipo pipeline che preveda vari stadi per eliminare i rumori e le caratteristiche del discorso tipiche dell'individuo, riconoscere gli elementi "puri" del discorso e interpretarli come sillabe, parole, frasi. Ad ogni stadio è necessaria un'elaborazione in parallelo ad alto livello per poter esaminare contemporaneamente un certo numero di possibili interpretazioni. Il riconoscimento del parlato

continuo e l'interpretazione del linguaggio naturale sono strettamente correlati; gli "indizi" forniti dal meccanismo di interpretazione sono importantissimi per il processo di riconoscimento. Gran parte dei suoni possono essere riconosciuti con un qualche grado di certezza solo se il loro contesto è noto (come talvolta accade nei discorsi tra uomini); un approccio ciclico può essere necessario per poter ipotizzare delle parole partendo dal riconoscimento di sillabe e quindi controllare se viene così creato un contesto che abbia senso.

9.4 Riconoscimento del linguaggio naturale

Il retroterra degli studi sul riconoscimento del linguaggio naturale è stato in parte trattato nel paragrafo 2.6. I due aspetti del problema affrontato in quel paragrafo sono la sintassi — il riconoscimento di strutture linguistiche — e la semantica — la ricerca del significato. Le ricerche in corso sul riconoscimento del linguaggio naturale ne aggiungono un terzo, il livello contestuale definito, in alcune opere, *copioneo script* (Hendrix e Sacerdoti, 1981). Questo approccio ammette che il riconoscimento del linguaggio non è possibile in una situazione non delimitata e che ogni applicazione in cui l'elaboratore interagisca con il suo utente attraverso il linguaggio naturale si svolge in un particolare contesto. I copioni o costruzioni simili definiscono i parametri generali del contesto e permettono che il significato delle affermazioni venga dedotto in relazione a questi contesti. Ad esempio, una base dati intelligente utilizzata per indagini criminali ha un particolare contesto, e tutte le affermazioni fatte sia dal computer che dai suoi utenti sono espresse in base a questo contesto.

Il riconoscimento del linguaggio naturale è un compito molto adatto per quasi tutti gli aspetti dello sviluppo dei computer della quinta generazione. Sia la sintassi che la semantica dei linguaggi naturali possono essere espresse in base a regole logiche; essi diventano così adatti alla programmazione in linguaggi basati sulla logica. Nella sua attività sul riconoscimento del linguaggio naturale, il gruppo dell'Icot sta seguendo questo metodo (Yasykawa, 1983). Gran parte dell'elaborazione, sia a livello sintattico che a livello semantico, comporta delle ricerche ad alta velocità su vaste basi di conoscenza. Questo tipo di operazione è adatta all'elaborazione in parallelo per mezzo di processori ad accoppiamento lasco in un'architettura a flusso o a riduzione dei grafi. A un livello inferiore, le operazioni di identificazione di

corrispondenze possono essere svolte nel modo migliore utilizzando i consueti chip VLSI (Allen, 1983). Di conseguenza, è ora possibile concepire una macchina per il riconoscimento del linguaggio naturale costruita da questi elementi hardware e software e realizzare così il meccanismo di interfaccia utente intelligente di un sistema di elaborazione della quinta generazione. La costruzione di tale macchina è, comunque, prevista appena per lo stadio finale del programma giapponese sulla quinta generazione ed è improbabile che qualche altro gruppo di ricercatori riesca a metterla a punto prima di quella data.

Col progredire delle ricerche sul riconoscimento e sull'elaborazione del linguaggio naturale, molti ricercatori hanno incominciato a raffreddare gli animi (per esempio, Kaplan e Ferris, 1982). Essi sottolineano che il linguaggio naturale, in molte situazioni, può essere un mezzo di comunicazione tra l'utente e il computer meno efficace di quanto possa sembrare. Il linguaggio naturale è molto ricco e sottile e può conferire significato a più livelli contemporaneamente. Però può essere anche vago, ambiguo, incompleto e inefficace. Molti critici contemplano l'idea di poter utilizzare il linguaggio naturale quale massimo linguaggio di programmazione, ma questo va contro la tendenza a realizzare linguaggi funzionali sempre più precisi, come spiegato nel capitolo 7. I problemi che sorgerebbero per dimostrare la correttezza di un programma scritto in un linguaggio naturale (il che richiederebbe che la dimostrazione stessa venga fatta in un linguaggio naturale) sarebbero quasi certamente insuperabili.

Il risultato più probabile è un compromeso tra diversi fattori. Da un lato, è improbabile che qualsiasi sistema di elaborazione sia in grado di gestire l'intera gamma e complessità di un linguaggio naturale nella sua totalità. Ma varcata una certa soglia, il sottoinsieme che si può gestire sarà abbastanza grande da essere utile in molte situazioni. Dall'altro lato questi vincoli possono diventare utili in ogni applicazione dato che forniscono dei limiti alle possibili interpretazioni di ogni affermazione e aiutano così ad evitare che in un sistema basato sulla conoscenza vengano assimilate cose insensate. In moltissime situazioni gli scambi verbali avvengono in base a un sottoinsieme ben definito del linguaggio naturale: basti pensare a pochi esempi ovvi, quali le operazioni chirurgiche, le transazioni del mercato azionario, il controllo del traffico aereo, il pilotaggio di una nave o di un sottomarino e la maggior parte delle situazioni in cui si svolge un allenamento. Si può immaginare, pertanto, che un sistema della quinta generazione sarà in grado di gestire un sottoinsieme del linguaggio utilizzato in un particolare contesto.

9.5 Elaborazione dell'immagine

L'elaborazione dell'immagine è considerata una tecnica di interfaccia essenziale per molte applicazioni della quinta generazione, in particolare per robot intelligenti, missili teleguidati, veicoli teleguidati per l'esplorazioni della superficie dei pianeti, sistemi CAD intelligenti, applicazioni che prevedono l'uso di mappe e molte applicazioni in campo medico. Come nel caso del riconoscimento della voce, è improbabile che si riesca a raggiungere il livello delle prestazioni umane: l'occhio umano ha 120 milioni di bastoncelli, 6,5 milioni di coni e 1 milione di filamenti nel nervo ottico (Duff, 1982). L'obiettivo dei gruppi di sviluppo della quinta generazione è sviluppare sistemi di elaborazione dell'immagine che siano sufficientemente discriminanti da risultare utili nelle applicazioni menzionate sopra. Le conoscenze di fondo per l'attività sull'elaborazione dell'immagine sono descritte nel paragrafo 2.7.

L'obiettivo dell'elaborazione dell'immagine è "riconoscere" gli oggetti presenti su uno schermo visivo come quelli presentati da un proiettore cinematografico e generare immagini della stessa qualità e con lo stesso senso della realtà. Come nel caso del riconoscimento del parlato, l'elaborazione dell'immagine opera a diversi livelli. Al livello più basso vi è un mezzo per identificare varie zone di un'immagine come appartenenti a forme particolari. I livelli intermedi riguardano l'identificazione delle forme in oggetti riconoscibili e il livello superiore è costituito da una base di conoscenza contenente le caratteristiche particolari delle forme che, quando richiesto, permette di elaborarle.

Gran parte delle ricerche sull'elaborazione dell'immagine riguarda lo sviluppo di hardware specializzato, soprattutto per il riconoscimento e la produzione delle forme. L'impostazione che sembra più promettente è costruire una matrice di elementi processori specializzati per ogni pixel dello schermo. Ogni processore è collegato agli otto processori vicini e ai meccanismi di controllo generale. Si veda la figura 9.3. Un'immagine della stessa qualità dell'immagine televisiva richiederà circa 1 milione di questi elementi di elaborazione; per i sistemi attualmente usati nella ricerca possono bastarne molti meno. Il sistema Clip4 dell'Università di Londra (Duff, 1982), ad esempio, usa una matrice di 96 per 96 processori. Varianti all'approccio della logica cellulare prevedono matrici sistoliche (paragrafo 5.8) di processori collegati in pipeline che non sono accoppiate in modo altrettanto stretto ed operano con un maggior grado di autonomia. Una tecnica promettente per la

produzione di immagini realistiche è l'uso di una classe di funzioni matematiche chiamate frattali. Queste prendono una forma elementare e la duplicano in versioni sempre più piccole all'interno di se stessa. Con leggere variazioni i frattali producono forme e tessiture strettamente somiglianti alla superficie e ai contorni degli oggetti naturali quali montagne, nuvole e corpi d'acqua.
Il livello più alto di elaborazione dell'immagine sopra citato richiede una struttura cognitiva per poter rappresentare ed elaborare simboli visivi. Questo determina un contesto per le operazioni più dettagliate di elaborazione dell'immagine, e una struttura di rappresentazione dell'immagine per la base di conoscenza che accompagnerà ogni applicazione. Per ora non c'è ampio accordo sul modo in cui si dovrà costruire questa struttura. Una linea di sviluppo utilizza quale modello l'architettura della corteccia visiva umana (McCormick *et al.*, 1982). Se questo sviluppo avrà successo, potrà forse essere applicato in altri campi dell'intelligenza artificiale.
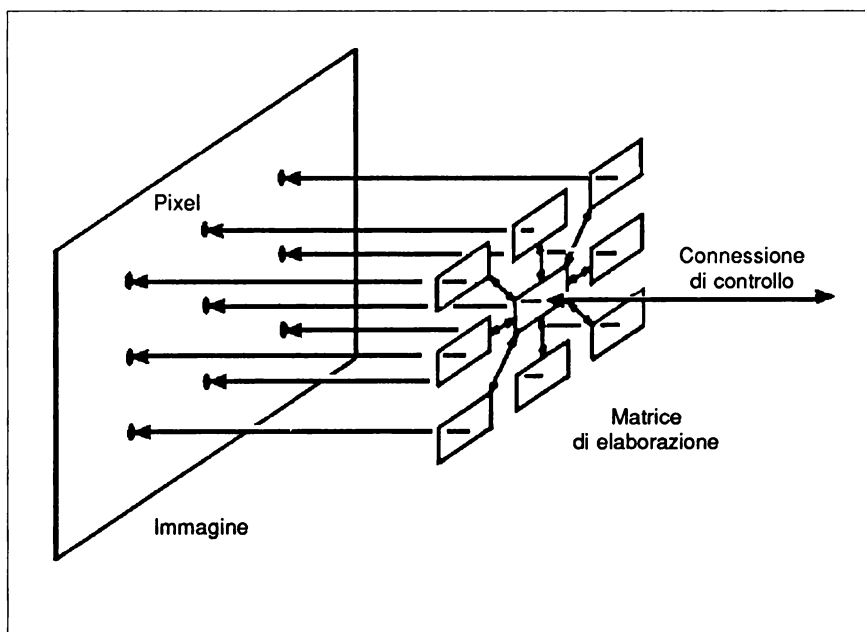
Fig. 9.3 Una matrice di processori di immagine.

9.6 Psicologia dell'interazione uomo-computer

Alla base delle nuove tecniche di interazione uomo-computer ci dovrebbe essere un patrimonio di conoscenza che descriva la struttura psicologica all'interno della quale si svolge questa interazione. L'approccio ingegneristico all'elaborazione dati prevalso negli ultimi quarant'anni sta proprio a indicare che non esiste attualmente un tale patrimonio di conoscenze. Comunque, la situazione cambierà grazie alle ricerche in corso presso diversi centri, per studiare gli aspetti del comportamento umano che intervengono nell'interazione tra computer e utente. La maggior parte delle interfacce utente della quarta generazione presentate nel paragrafo 9.2 sono il risultato dei primi stadi di questa attività.

Il problema più ovvio, presente fin da quando i primi computer venivano programmati e controllati da pannelli di comando, è il divario cognitivo tra l'utente e l'elaboratore. Sono stati individuati quattro aspetti principali di questo problema (Harris, 1982): il problema del linguaggio, problemi conseguenti al punto di vista concettuale dell'utente, il problema di districarsi nei percorsi di controllo di un programma e le differenze tra le filosofie insite nelle interfacce utente dei diversi pacchetti applicativi. Il problema del linguaggio è dovuto, tra i vari motivi, al gergo utilizzato dai progettisti dei sistemi di elaborazione nelle interfacce utente, e all'uso particolare che viene fatto del linguaggio naturale all'interno della comunità degli informatici (un caso classico sono i nomi pluricomposti, in inglese utilizzati come aggettivi: "high resolution graphics system memory overflow error address" [=indirizzo di errore di straripamento di memoria dei sistemi ad alta risoluzione grafica] è un chiaro esempio.) Per quanto la situazione sia molto migliorata negli ultimi anni, in molti programmi i percorsi di controllo sono lunghi e tortuosi ed è pertanto piuttosto facile che l'utente si perda. Inoltre, molti programmi non hanno dei semplici meccanismi di uscita. La differenza nella filosofia dell'interfaccia utente tra i vari programmi è colmata, in certa misura, da svariati pacchetti software integrati e dalla ricerca, da parte di certi produttori di hardware, che tutto il software immesso nei loro elaboratori sia conforme a certi principi dell'interfaccia utente.

Sono in corso due linee di ricerca che dovrebbero essere di grande aiuto nella progettazione di interfacce utente intelligenti. Una è un tentativo di identificare quelle capacità di comunicazione che rendono alcuni specialisti in grado di comunicare la loro esperienza agli altri in modo molto efficace. Se risulterà possibile incorporare alcune di queste capacità di comunicazione nella struttura

delle interfacce utente intelligenti o nella filosofia alla base della loro progettazione, allora la qualità di queste interfacce migliorerà notevolmente. L'altra linea di ricerca è un'analisi del comportamento umano in situazioni che prevedono la risoluzione di problemi complessi: quale componente dello sforzo mentale deduttivo, induttivo, laterale, euristico o intuitivo entra effettivamente in gioco quando un esperto si accinge a svolgere un compito difficle? Quando si avranno risposte a questo problema (e certamente saranno risposte diverse da un settore specialistico all'altro), si potranno fare dei seri tentativi per ridurre il divario cognitivo tra l'utente e l'elaboratore.

9.7 Conclusione

Il successo o il fallimento dell'iniziativa per i computer della quinta generazione dipende per molti aspetti dai progressi compiuti nelle interfacce utente intelligenti. I computer della quinta generazione non sono destinati a sostituire l'intelligenza umana; essi comunque richiedono un mezzo di comunicazione con il loro utente di livello superiore a quello attuale per rendere accessibile e quindi produttiva la loro intelligenza. Stabilire la comunicazione a livello adeguato tra un computer e il suo utente è parte essenziale e integrante dei sistemi intelligenti basati sulla conoscenza. I computer della quinta generazione non sono destinati ad essere il privilegio di pochi, ma a portare i benefici della tecnologia informatica avanzata a tantissime persone di qualsiasi estrazione e di qualsiasi paese. Questo obiettivo sociale è esplicito nel programma per la quinta generazione giapponese ed è un requisito essenziale per una prospera esistenza dell'industria informatica e dei progettisti di sistemi di tutto il mondo, dato che essi dipendono sempre più dalla vendita dei loro prodotti su ampi mercati. Se non fossero facilmente accessibili e utilizzabili, i computer dotati di maggiore intelligenza potrebbero facilmente essere guardati con sospetto e paura, o semplicemente essere ignorati da gran parte dei loro potenziali utenti.

10

Applicazioni dei computer della quinta generazione

Parlando delle possibili applicazioni dei computer della quinta generazione si deve tener conto del tempo necessario alla loro realizzazione. Il completamento delle prime implementazioni della nuova tecnologia è previsto per l'inizio degli anni novanta, ed è piuttosto difficile prevedere quali saranno i dispositivi di elaborazione necessari in quel lontano futuro. Si prevede, comunque, che i risultati intermedi delle ricerche sulla quinta generazione porteranno a nuove o migliori applicazioni prima del completamento dell'opera principale. I programmi della quinta generazione traggono la propria forza soprattutto dal voler raggiungere certi obiettivi — elaboratori più intelligenti — anziché dalla esplicita richiesta di realizzare nuove applicazioni specifiche. Il concetto fondamentale dei computer della quinta generazione comprende un'interfaccia utente intelligente che li renda adatti ad essere utilizzabili da un vasto gruppo di persone, caratteristica, questa, molto importante dati gli aspetti sociali di questi sviluppi.
Poste queste condizioni, è tuttavia possibile delineare, con un certo grado di sicurezza, le probabili aree di applicazione dei computer della quinta generazione. Alcune, quali i sistemi esperti, sono un'estrapolazione degli sviluppi attuali, mentre altre, come gli elaboratori di testi attivati dalla voce, sono completamente nuove. Il ruolo primario dei computer della quinta generazione è, comunque, essenzialmente la risoluzione di problemi; la metodologia di lavoro è l'inferenza logica; e fra le applicazioni sono comprese aree di giudizio e discernimento (Sell,

1982). In questo capitolo le aree di applicazione vengono raggruppate per attività: applicazioni industriali, militari, commerciali, di progettazione e didattiche, sistemi esperti. L'intento non è presentare un elenco esaustivo e neppure avanzare esatte previsioni sul futuro, ma trattare alcune possibili applicazioni dei computer della quinta generazione per poter illustrare concretamente i prodotti finali, i cui elementi sono stati descritti nei capitoli precedenti.

10.1 Applicazioni industriali

L'attuale generazione di robot industriali ha determinato una rivoluzione nel mondo industriale, in particolare nell'industria automobilistica. Da un lato sono stati d'importanza decisiva per le case costruttrici, aumentando la loro produttività; dall'altro hanno portato alla disoccupazione migliaia di operai. I robot della prima generazione hanno prestazioni molto limitate: ciascuno, infatti, può svolgere solo pochi compiti e ciascun compito deve essere programmato come una sequenza ben precisa di operazioni. Le ricerche sulla prossima generazione di robot hanno preceduto l'iniziativa della quinta generazione, ma allo stesso tempo, in certa misura, sono state assimilate in essa. L'obiettivo è produrre robot con un'intelligenza di gran lunga maggiore e riuscire a sviluppare capacità sensoriali tattili e visive.
Un robot della nuova generazione sarà in grado di svolgere una vasta gamma di compiti all'interno di una struttura generica di scopi e obiettivi di cui è fornito. Sarà dotato della capacità di autoapprendimento e di un'interfaccia per la comunicazione ad alto livello con il suo "pensatore". Ad esempio, fornendogli gli obiettivi di un determinato compito, un robot di questo tipo sarà in grado di determinare la propria sequenza di passi procedurali in modo da svolgere quel determinato compito e di modificare quei passi secondo le esigenze esterne. Esso potrebbe, ad esempio, sviluppare una sequenza per eseguire un compito nel minor tempo possibile, e un'altra per usare meno energia possibile. Sottoponendo continuamente a controllo la propria prestazione, potrebbe perfezionare le proprie sequenze alla luce delle esperienze precedenti.
I robot vengono spesso installati come parte di sistemi integrati di produzione flessibile, in cui tutti gli aspetti del processo sono sotto il controllo centralizzato di un computer. È probabile che l'impiego di robot intelligenti, unitamente a innovazioni dello stesso tipo apportate all'intero sistema di controllo della produzione,

sarà un progresso tanto importante nelle tecniche industriali quanto la prima introduzione dei robot. È anche probabile che avrà gli stessi effetti sull'occupazione in questo settore.

10.2 Applicazioni militari

Quello delle applicazioni militari è uno dei settori più ampi in cui vengono utilizzati i computer della quinta generazione. Si possono individuare due classi di applicazioni: i sistemi che svolgono prevalentemente un servizio di consulenza e quelli quasi completamente automatici (Taylor, 1983). Nella seconda categoria rientrano i sistemi di armamenti di tutti i tipi nei quali viene incorporato un maggior grado di intelligenza. Il missile Cruise terra-terra rappresenta lo stato dell'arte degli attuali sistemi di controllo. Esso ha programmate al suo interno le mappe, rappresentate in forma digitale, del tragitto che deve percorrere e la localizzazione dei suoi obiettivi. Segue quindi una rotta predeterminata adattando la propria altitudine alle mappe incorporate ed alle informazioni ricevute dal suo radar di perlustrazione del terreno. Un missile Cruise intelligente sarebbe in grado di variare il proprio corso in base alle condizioni incontrate lungo il tragitto, di adottare misure di evasione contro i sistemi di difesa missilistici e (magari) di scegliere obiettivi alternativi in base alle decisioni tattiche prese durante il volo. Data la natura nucleare delle testate dei missili Cruise, è probabile che quest'ultima possibilità sarà motivo di profonde controversie. Sistemi di guida "intelligenti" dello stesso tipo, verranno certamente incorporati, in misura variabile, nei missili antiaerei, nei missili antinave e nei missili balistici intercontinentali.
I sistemi di difesa strategica proposti, attualmente a livello di studio di fattibilità, contano interamente su sistemi di computer intelligenti per eseguire le proprie attività. Essi propongono di utilizzare una combinazione di armamenti con base su satelliti o a terra, basati su fonti di raggi laser e su altre fonti di radiazioni ad alta energia, per distruggere numerosi missili in volo. Si ritiene che alcuni dei loro requisiti superino persino la portata dei computer della quinta generazione: ad esempio, come gestire, in tempo reale, un vero attacco ICBM, in cui ogni missile in arrivo ha testate multiple, ciascuna con obiettivi propri, e libera in volo parecchie testate fittizie? Se i sistemi di missili antibalistici di questo tipo risulteranno fattibili, dovranno essere dotati di sistemi computerizzati in grado di prendere numerose ed

importanti decisioni strategiche, tattiche e operative in tempi molto brevi, e in grado di eseguire un numero enorme di calcoli necessari per individuare i missili in arrivo e controllare le misure di difesa.
I sistemi di comando, controllo e comunicazione militari diventano sempre più computerizzati. Se saranno disponibili dei computer con un grado maggiore di intelligenza, si potranno realizzare le nuove generazioni di questi sistemi. Probabilmente molti di questi sistemi saranno attivati dalla voce, come avviene già per i sistemi di controllo dei sottomarini e per altri sistemi simili. La primissima applicazione degli elaboratori elettronici — scoprire i codici delle comunicazioni nemiche — potrà migliorare notevolmente se i computer saranno in grado di operare con il linguaggio naturale. I sistemi interattivi di supporto alle decisioni tattiche ed operative contribuiranno a svolgere l'attività di comando degli ufficiali a vari livelli. La guerra elettronica — in cui ogni parte cerca di intercettare e interrompere i sistemi di comando e di comunicazione dell'altra — assumeranno una nuova dimensione quando verranno introdotti computer capaci di prendere decisioni intelligenti a velocità elettroniche. La guerra aerea e sottomarina viene condotta in misura sempre crescente con questi metodi, e non mancano attività di ricerca volte alla realizzazione di nuovi sviluppi in questi settori.

10.3 Applicazioni commerciali

Per quanto la realizzazione del concetto di ufficio elettronico sia più lontana di quanto si fosse previsto, il movimento verso l'automazione degli uffici è ormai un fenomeno irreversibile e in continuo sviluppo. La meta attuale è la realizzazione di reti integrate di comunicazione e di computer per uffici in cui tutti gli aspetti di un'azienda o di una società — produzione, controllo di magazzino, vendite, acquisti, stipendi e contabilità — vengano gestiti da un insieme di sistemi elettronici collegati. È probabile che i computer della quinta generazione consentiranno di costruire un ulteriore livello al di sopra dei database commerciali integrati di questo tipo. Le informazioni operative nel database verranno a costituire la materia prima di una base di conoscenza commerciale. Su questa base di conoscenza verranno costruiti dei sistemi di supporto decisionale per i manager, così come dispositivi di analisi approfondita che permettano di effettuare valutazioni strategiche sulle informazioni commerciali. Queste saranno utili nello stanziamento delle risorse,

nella programmazione delle scadenze, nelle analisi di mercato e nella pianificazione della società.
Gli elaboratori di testi attivati dalla voce costituiscono il parametro più valido per valutare concretamente l'eventuale successo dei programmi di sviluppo dei computer della quinta generazione. Essendo l'elaborazione dei testi una delle applicazioni oggi più diffuse, poter dettare direttamente a un elaboratore di testi sarebbe un grossissimo vantaggio (oltre a essere un'allettante prospettiva di mercato per le società capaci di sviluppare sistemi di questo tipo). Un'applicazione correlata sono i sistemi automatici di traduzione delle lingue. In teoria questi dovrebbero essere completamente automatizzati, ma in pratica è probabile che essi dovranno interagire con un operatore esperto che si occupi dei casi che l'elaboratore non riesce a gestire. La combinazione dei due sistemi permetterà, ad esempio, di produrre contemporaneamente le relazioni degli incontri internazionali in tutte le lingue dei partecipanti. Documenti commerciali, d'ingegneria, scientifici, potrebbero essere dettati da una parte ed essere trascritti e trasmessi ad un'altra parte e qui essere tradotte automaticamente, in tempo brevissimo. Elaboratori di testi attivati dalla voce e i sistemi di traduzione automatica delle lingue sono oggetto di sviluppo in Gran Bretagna, come progetti di dimostrazione nel programma Alvey, e in Giappone.

10.4 Applicazioni per la progettazione

La progettazione assistita dall'elaboratore è un'area di applicazione con una storia lunga e movimentata. I sistemi CAD stanno iniziando ora a mantenere le loro precedenti promesse. Essi costituiscono un aspetto fondamentale nella progettazione di veicoli a motore, navi, aereoplani e veicoli spaziali, e sono l'unico mezzo di sviluppo dei moderni circuiti integrati. Saranno i sistemi CAD della quinta generazione a occuparsi sempre più del processo di effettiva progettazione, sia che si tratti del componente di un'automobile che di un chip, e ad interagire con il loro operatore a livello di specifiche di progetto. Dati i parametri generali di prestazione di un prodotto, e fornite alcune direttive e alcune limitazioni, i sistemi CAD della nuova generazione metteranno a punto sia i dettagli che li riguardano, basati su database di progetto molto ampi e complessi, sia le basi di conoscenza della metodologia di progetto. Come già spiegato nel paragrafo 5.9, i sistemi CAD di

questo tipo sono essenziali nella progettazione degli stessi computer della quinta generazione; si prevede che ci sarà un andamento di sviluppo ciclico in cui i risultati raggiunti nei sistemi di elaborazione verranno utilizzati come ausilio nella progettazione della fase successiva.
Si prevedono risultati simili anche nei sistemi di supporto alla progettazione per il software (paragrafo 6.5). Per quanto l'obiettivo di un Ipse sia meno ambizioso della progettazione automatica da specifica, esso è destinato a svolgere un ruolo molto importante nella generazione, a costi accessibili, di un codice che risulti corretto e in accordo con le specifiche. È probabile che insieme, i sistemi CAD per l'hardware e gli Ipse per il software, costituiranno dispositivi per la progettazione dei sistemi della quinta generazione integrati utilizzabili per una vastissima gamma di prodotti basati sull'elettronica.

10.5 Applicazioni didattiche

L'istruzione assistita dal calcolatore (CAL, *Computer-Assisted Learning*) è un'area di applicazione in cui molte delle aspettative originali sono andate deluse. L'interazione tra l'insegnante e lo studente è molto più sottile e complessa di quanto avessero immaginato i progettisti dei primi sistemi CAL. L'avvento dei computer dotati di un maggior grado di intelligenza, operanti su basi di conoscenza anziché su basi di informazioni, potrebbe portare a notevoli progressi; è comunque improbabile che il ruolo centrale dell'insegnante venga soppiantato da un computer. Si potrebbero, però, mettere a punto sistemi per l'insegnamento di pratiche elementari, capaci di diagnosticare, analizzare, e aiutare a correggere gli errori dello studente. Una certa attività preliminare viene attualmente svolta sui principi di progettazione di tali sistemi. Questi non sono basati sui tradizionali questionari a difficoltà crescente, ma su domande orientate a obiettivi, valutazioni e giudizi basati su regole. Si sta lavorando anche su sistemi in grado di costruire modelli corrispondenti ai modi tipici in cui gli studenti sbagliano, e di utilizzare questi stessi modelli per aiutarli a correggere i propri errori.
Un punto meno controverso è l'evoluzione dai database didattici alle basi di conoscenza che siano accessibili per mezzo di vasti sottoinsiemi di un linguaggio naturale. Queste dovrebbero aiutare a superare la riluttanza di molti pedagogisti all'impiego dei computer e permettere agli insegnanti ed agli studenti di accedere

a memorie di massa di grandi dimensioni contenenti informazioni altamente strutturate.

10.6 Sistemi esperti

L'area di applicazione dei computer della quinta generazione con maggiori prospettive è costituita dai sistemi esperti. Per molti anni i sistemi esperti hanno costituito un aspetto importante delle ricerche sull'intelligenza artificiale, e qualche sistema è stato anche utilizzato per un certo periodo (il primo risale al 1965). Hanno già superato il momento del primo impatto, ma l'avvento dei computer della quinta generazione dovrebbe fornire il mezzo per diffonderli su vasta scala.
Come detto nel paragrafo 2.8, i sistemi esperti sono il prodotto dell'applicazione delle tecniche dell'intelligenza artificiale a settori specifici. Un sistema esperto è un sistema di elaborazione che ha assimilato parte delle conoscenze di un esperto umano, come un medico, un geologo, un chimico analitico o un agente di borsa. Alcuni sistemi esperti sono destinati a essere utilizzati quali consulenti interattivi dagli esperti dello stesso settore, altri a essere utilizzati da personale meno qualificato o non specializzato. Dato un problema nel suo campo specialistico, un sistema esperto dovrebbe essere in grado di risolverlo e, se necessario, di fornire all'utente una spiegazione della sua linea di ragionamento. Molti operano in maniera interattiva, permettendo, da un lato, che l'utente fornisca le informazioni in maniera flessibile e, dall'altro, fornendo essi stessi le conclusioni intermedie non appena queste siano state tratte. In alcune applicazioni, la conoscenza specialistica è "fissata" nel sistema quando questo viene creato. In altre il sistema è in grado di apprendere dalla sua stessa esperienza (anche dai suoi errori). Attualmente sono disponibili due tipi di sistemi esperti: quelli che hanno già "dentro" una certa competenza in un settore specifico e gusci di sistemi esperti che hanno dispositivi generali di elaborazione della conoscenza, ma in cui, per poter avere un sistema funzionante, la competenza specifica deve essere fornita dalla cooperazione di un ingegnere della conoscenza con un esperto.
Tutti i sistemi esperti operano su una base di conoscenza che generalmente viene aggiornata via via che viene trattato un nuovo caso e, per la maggior parte, hanno insiemi di regole che esprimono i "segreti del loro mestiere". Sono dotati di teniche di controllo per l'applicazione delle regole alla base di conoscenza, nella risoluzio-

ne dei problemi che gli vengono posti (paragrafo 2.3). I primissimi sistemi esperti erano confinati a settori in cui la conoscenza è ben strutturata, secondo i seguenti criteri: spazio di ricerca ridotto, base di conoscenza valida e priva di contraddizioni, dati affidabili e statici forniti dall'utente durante l'interazione con il sistema esperto (Alty e Coombs, 1984). I sistemi più recenti sono riusciti ad allentare notevolmente questi vincoli permettendo così di costruire sistemi esperti in campi sempre più complessi. I computer della quinta generazione dovrebbero permettere di sciogliere questi vincoli quasi completamente, rendendo così possibile progettare un sistema esperto utilizzabile in un campo in cui la base di conoscenza sia molto vasta, la conoscenza sia vaga, incompleta e contradditoria e lo stato di un problema possa cambiare durante l'analisi da parte dello stesso sistema esperto. Molti problemi del settore medico rientrano in quest'ultima categoria: un computer che assista il chirurgo e collabori con lui in tempo reale nel corso di un'operazione complessa non è che un esempio di ciò che si potrebbe realizzare quando i computer della quinta generazione saranno in grado di ospitare dei sistemi esperti.

Tra le aree di competenza trattate dai primi sistemi esperti figurano: diagnosi e terapia delle malattie infettive (Mycin, Stanford University), prospezione geologica (Prospector, SRI International/US Geological Survey), medicina interna generale (Internist, Pittsburgh University), determinazione di strutture chimiche sulla base dei dati di uno spettrometro di massa (Dendral, Stanford University, il primo sistema esperto operativo), configurazione di sistemi di microcomputer (R1, Digital Equipment Corporation) e progettazione di esperimenti genetici (Molgen, Stanford University). È probabile che i sistemi esperti nel settore medico rimarranno un importante settore di sviluppo — l'obiettivo di un assistente che aiuti il medico generico a formulare la diagnosi viene perseguito attivamente presso vari centri. Uno dei progetti dimostratori del programma Alvey è un sistema esperto da utilizzarsi nelle richieste di indennità alla previdenza sociale, visto che il sistema di previdenza sociale inglese è pressoché impossibile da capire senza l'ausilio di un esperto di questo tipo. Tra gli altri possibili utilizzi dei sistemi esperti vi sono vari settori legislativi, la progettazione di reti stradali e ferroviarie, la tassazione, l'analisi di mercato, le previsioni metereologiche, la gestione finanziaria e numerosi sistemi militari, tattici e strategici.

10.7 Conclusione

Va sottolineato che a questo punto non è possibile anticipare con certezza quali potrebbero essere le future applicazioni specifiche dei computer della quinta generazione. Definito chiaramente il concetto di computer della quinta generazione, è chiaro quali saranno le aree di applicazione, e sono stati identificati i gruppi di potenziali utenti. La natura ed il tempo di realizzazione delle specifiche applicazioni dipendono dall'ordine con cui vengono risolti alcuni dei problemi di base delle ricerche sulla quinta generazione quali il riconoscimento della voce, l'elaborazione dell'immagine e, soprattutto, la rappresentazione della conoscenza in generale. È probabile che si arriverà alla fine dei programmi di ricerca sulla quinta generazione attraverso una sequenza ciclica di sviluppi; i prodotti disponibili quando un ciclo di ricerche sarà completato saranno utilizzati nello sviluppo degli stadi successivi. Resta ancora da vedere se si raggiungerà mai una resa commerciale che compensi tutti gli investimenti fatti sull'attività di ricerca e sviluppo per la quinta generazione.

11

Prospettive della quinta generazione

A questo punto è impossibile dire con certezza in quale misura si riusciranno a realizzare gli obiettivi della quinta generazione. Molte conseguenze di quest'iniziativa dipenderanno da quali obiettivi verranno conseguiti, da quali gruppi di sviluppo e in quali paesi, e dal modo in cui questi sviluppi verranno sfruttati. Nella maggior parte di questo capitolo si parte dal presupposto che questi obiettivi verranno realizzati, almeno nella misura in cui i sistemi intelligenti basati sulla conoscenza sono una proposta realistica. Il problema di stabilire la probabilità di successo di tutto questo schema di sviluppi viene rimandato all'ultimo paragrafo.
I sistemi di elaborazione della quinta generazione appartengono al più ampio campo della tecnologia dell'informazione, che comprende anche i sistemi di controllo elettronici e le telecomunicazioni. La tecnologia dell'informazione è causa di cambiamenti, molto potente di per se stessa, ma è solo uno dei tanti punti di pressione tecnologica: altri sono le ricerche biochimiche, l'ingegneria genetica e l'impiego ulteriore, a scopi pacifici o meno, dell'energia nucleare. Le stesse innovazioni tecnologiche, a loro volta, non sono che una delle tante forze che esercitano una certa pressione in favore di ulteriori innovazioni. Altre forze sono quelle di natura religiosa, politica, economica, pressioni derivanti dall'espansione demografica e dalla diminuzione delle risorse e pressioni ambientali, conseguenza del progressivo avvelenamento del nostro piccolo pianeta per effetto di sostanze inquinanti. A distanza di brevi periodi, tutto questo sfocia in guerre, attacchi

terroristici, carestie, scioperi, proteste o nell'estinzione di qualche altra specie animale o vegetale, fenomeni che nessun computer intelligente o miracolo di ingegneria genetica può scongiurare. La tecnologia dell'informazione opera nella stessa area delle altre forze che causano dei mutamenti, interagisce con esse e comporta conseguenze tanto complesse e di difficile comprensione quanto la rinascita dell'Islam.
È in questo contesto che vanno valutate le conseguenze dello sviluppo dei sistemi intelligenti. Le conseguenze potrebbero essere di vasta portata e saranno sentite da individui, società e nazioni in maniera diversa.

11.1 L'impatto dei sistemi intelligenti basati sulla conoscenza

Fin dall'inizio, il ruolo centrale di tutti i tipi di sistemi di elaborazione delle informazioni è stato quello di un'importante forza trainante nello sviluppo della nuova generazione di computer. Questo è, in particolare, il caso del Giappone: "I sistemi di elaborazione delle informazioni sarannno strumenti fondamentali in tutti i settori di attività sociale tra cui l'economia, l'industria, l'arte e la scienza, l'amministrazione, le relazioni internazionali, l'istruzione, la cultura e la vita quotidiana ecc." (JIPDEC, 1981). I sistemi di elaborazione intelligenti avranno un impatto notevole, talvolta anche drammatico, in tutti questi settori. Ne citiamo alcuni esempi qui si seguito.
Il campo medico è probabilmente il più utile patrimonio di conoscenze ed esperienze attualmente esistente. Esso ha tutte le caratteristiche distintive della quinta generazione: un corpus vastissimo, basato su un insieme di principi scientifici ben stabiliti, ipotesi di lavoro, regole empiriche ed esperienze raccolte ma non ancora formalmente convalidate. Ha anche lacune molto grosse e ambiti in cui le conoscenze sono vaghe e talvolta contraddittorie. Ciò nonostante la maggior parte delle persone attualmente viventi devono la loro salute o addirittura la loro vita alla medicina moderna e ai suoi specialisti. Se i computer della quinta generazione saranno in grado di assimilare vaste porzioni delle attuali conoscenze mediche, e di fornirle a sistemi esperti in grado di migliorare e aumentare le capacità dei medici, a tutto il mondo potrebbero derivare notevoli benefici. Molti paesi del Terzo Mondo hanno una carenza endemica di medici e basano i loro servizi di sanità rurali su personale paramedico della tradizione dei "medici scalzi" cinesi. Se a questo

personale paramedico si potesse fornire qualche piccolo strumento che faciliti la formulazione della diagnosi e che sia costruito saldamente, adatto ai loro ambienti di lavoro e comandato dalla voce, la qualità dei loro servizi di base migliorerebbe notevolmente. Questo, a sua volta, determinerebbe un notevole miglioramento nei servizi sanitari di questi paesi.

La gestione delle risorse naturali, sia quelle rinnovabili, come il legname, che quelle finite, come il petrolio, costituisce un problema di crescente importanza. La situazione è molto complessa e vanno considerati fattori geologici, ecologici, economici e politici. L'uso dei sistemi esperti e di supporto decisionale per delineare le politiche di organizzazione, e l'implementazione di queste stesse politiche, potrebbe portare a notevoli vantaggi. Uno dei maggiori problemi che la Gran Bretagna dovrà affrontare nei primi decenni del prossimo secolo rientra proprio in questa categoria: cosa fare in seguito all'impoverimento o nel caso di esaurimento delle riserve petrolifere del Mar del Nord.

In parecchi settori dell'amministrazione sociale, in particolare quello fiscale, i sistemi di indennità della previdenza sociale, la politica degli alloggi e dei trasporti, la situazione è diventata tanto complessa da essere quasi ingestibile. L'uso dei sistemi di elaborazione intelligenti, che utilizzano le conoscenze dedotte dalle basi di dati che già si stanno realizzando in questi settori, potrebbe aiutare a renderli più comprensibili ed efficienti, eliminando in gran parte errori, ingiustizie e ritardi che per anni sono stati accettati quale parte di queste burocrazie. Utilizzando i computer nell'amministrazione sociale, si corre sempre il pericolo di infrangere le libertà individuali: la quinta generazione, con la sua capacità di prendere decisioni intelligenti riguardanti dati personali o sociali, potrebbe far sorgere problemi di gran lunga maggiori rispetto a quelli creatisi in seguito all'introduzione dei computer tradizionali.

Nel mondo commerciale, tutti ormai ritengono che il profitto di un'organizzazione sia direttamente proporzionale alla qualità dei suoi sistemi informativi. Questo continuerà sicuramente ad essere vero quando saranno disponibili i sistemi di supporto decisionale intelligenti con basi di conoscenza annesse. La sopravvivenza o la scomparsa di molte aziende, grandi e piccole, dipenderà dal modo in cui gestiranno il passaggio alla tecnologia della nuova generazione. Le ricompense del successo saranno enormi: standard molto più alti di programmazione e gestione aziendale, ottimizzazione della produzione, del marketing, del controllo dei crediti e di altre pratiche commerciali. Comunque, nel clima economico dei prossimi dieci

anni, che probabilmente sarà almeno tanto duro quanto quello attuale, le aziende che non saranno in grado di competere ai livelli nazionali e internazionali non sopravviveranno.

Le conseguenze che i computer della quinta generazione avranno sull'occupazione saranno, probabilmente, tanto complesse quanto quelle dell'attuale tecnologia dell'informazione: creazione e supporto diretto di alcune attività nell'industria della tecnologia dell'informazione, creazione e supporto indiretto di numerosi posti di lavoro grazie al contributo a un aumento della produttività e perdita di occupazione, invece, laddove il lavoro venga svolto da un computer o da processi controllati da computer. Come sopra accennato, le società che prospereranno e quindi garantiranno un impiego sicuro ai propri dipendenti saranno quelle dotate di validi sistemi informativi. La differenza sarà che la soglia di sostituzione delle persone da parte dei sistemi elettronici sarà più alta — tra lo strato basso e medio dell'organizzazione. Le aziende saranno in grado di operare con un numero ridotto di responsabili, forse in nuove strutture manageriali, tutti operanti con l'ausilio di sistemi di supporto manageriale legati alla base di conoscenza della società.

È piuttosto probabile che l'avvento dei computer della quinta generazione coincida con, e sia parte integrante di un aumento generale degli standard e dalla qualità del lavoro in tutti i settori. I consumatori cominciano ad aspettarsi standard migliori di progettazione, qualità e sicurezza ed una diminuzione dei consumi energetici per ogni tipo di prodotto, e servizi più veloci, semplici ed efficienti sia per il settore pubblico, sia per quello privato. La diffusione dell'uso dei computer dotati di maggiore intelligenza accelererà notevolmente questo processo, che potrebbe comportare conseguenze sociali ed economiche di vasta portata.

11.2 Conseguenze militari

L'applicazione generale dei computer della quinta generazione in campo militare comporta serie conseguenze che sono diventate già argomento di discussione. La nuova tecnologia farà sì che si vada sempre più verso una maggior meccanizzazione delle guerre e che, per la prima volta, venga trasferito un certo potere decisionale per tattiche e strategie militari a sistemi assistiti dall'elaboratore, o a sistemi completamente automatizzati. Ad esempio, la maggior parte delle decisioni operative prese da un sistema di difesa strategica dovranno essere assunte auto-

maticamente data la velocità con cui il sistema deve reagire. Questo determina tre problemi: l'etica di delegare un computer a prendere decisioni militari, il pericolo di un conflitto iniziato inavvertitamente per un malfunzionamento del computer e la possibilità di una sconfitta militare devastante conseguente al malfunzionamento o ad un guasto di un sistema elettronico critico. Quest'ultimo aspetto è motivo di preoccupazione per molti esperti di tecnologia dell'informazione che operano in questo settore. Essi, infatti, sono convinti che non tutti gli errori latenti, in un sistema della complessità di quelli previsti, possano mai essere individuati e corretti.

11.3 Il mercato della tecnologia dell'informazionze

Il mercato della tecnologia dell'informazione non è mai stato un rifugio per i pusillanimi dato che sembra sempre alternare periodi di notevole ottimismo a periodi di depressione e confusione. Il mercato dei semiconduttori ha un andamento ciclico di carenza e sovraofferta a livello mondiale e i prezzi variano di conseguenza. Molti investitori considerano con riluttanza la possibilità di investire i propri capitali in un settore che, pur essendo in continua e rapida ascesa, è soggetto a tanti sovvertimenti. In un settore di tanto rapida innovazione, e, di conseguenza, di altrettanto rapida obsolescenza, la maggior parte dei prodotti sono fortunati se riescono a sopravvivere sul mercato per cinque anni. Dati i costi elevati dello sviluppo dei prodotti hardware e software, la differenza tra grossi profitti e grosse perdite spesso dipende da un piccolo numero di vendite.
La quinta generazione sembra destinata ad alzare ulteriormente la posta. Ci sarà uno spostamento dallo sviluppo, ad alta intensità di manodopera, di hardware e di software a tecniche basate su fonderie di silicio ed ambienti di sviluppo del software integrati, ad alta intensità di capitale. La notevole complessità dei sistemi della quinta generazione significa che, anche con gli attuali aiuti allo sviluppo, i costi per immettere sul mercato un nuovo prodotto saranno molto elevati. Comunque, se le interfacce utente intelligenti terranno fede alle loro promesse, il mercato per i prodotti della quinta generazione sarà molto più ampio al punto che, se verranno concessi adeguati fondi di sviluppo, si estenderà a tutto il Terzo Mondo. I tradizionali sistemi di elaborazione diventeranno presto obsoleti e verranno rimpiazzati da sistemi equivalenti dotati di maggiore intelligenza. Allo stesso modo c'è il pericolo che i professionisti della tecnologia dell'informazione che abbiano

maturato la loro esperienza nel settore informatico tradizionale facciano la stessa fine, e vengano sostituiti da ingegneri della conoscenza che sappiano utilizzare gli strumenti di progettazione dei sistemi della quinta generazione.
Come è già accaduto per le quattro precedenti generazioni, il piano della quinta generazione è principalmente uno sforzo nato dalla volontà di raggiungere certi obiettivi. L'iniziativa giapponese, che ha dato inizio a questa attività a livello mondiale, è frutto di considerazioni generali di natura sociale ed economica, ma non in particolare di precise esigenze di mercato. È stato più volte fatto notare che i computer sono una soluzione in cerca di un problema: la quinta generazione non fa eccezione. Tradurre i progressi teorici compiuti nell'elaborazione della conoscenza e nel riconoscimento della voce in prodotti pratici, commerciabili e redditizi è, a prima vista, un'impresa almeno tanto ardua quanto lo sviluppo degli elaboratori elettronici intelligenti. Supponendo che si compiano i progressi teorici necessari, sarà questo ostacolo finale a determinare, in ultima istanza, se tutto il tempo, la fatica e il denaro investiti nei programmi della quinta generazione saranno stati spesi bene.

11.4 La tecnologia dell'informazione di serie A

Nonostante i problemi a cui abbiamo accennato, la tecnologia dell'informazione sta diventando un settore industriale molto importante. È fonte di crescita e di stabilità economica, importante datore di lavoro, generatore di benessere e, costantemente impegnata in attività di ricerca e sviluppo, una valida garanzia di futura prosperità economica per il paese che la ospita. Sta anche diventando la chiave per la vitalità economica dei settori industriale, commerciale e sociale, dato che cominciano a dipendere dai sistemi elettronici in tutti gli aspetti delle loro attività. Di conseguenza, la forza e le dimensioni del settore industriale della tecnologia dell'informazione di ogni paese sviluppato costituiscono un aspetto di notevole importanza sia economica che politica. Dai primi tempi in cui gli Stati Uniti erano i maggiori fornitori di hardware e la Gran Bretagna il maggior produttore di software del mondo, nella serie A della tecnologia dell'informazione ci sono stati molti cambi di posto. La situazione si complica ulteriormente per la natura internazionale della maggior parte delle principali società di tecnologia dell'informazione e per il fatto che quasi tutti gli elaboratori sono un aggregato di componenti fatti in tutto il mondo. Alcuni chip, tra la fabbricazione del wafer e le

fasi di confezione, viaggiano addirittura da una parte all'altra del mondo.
È, comunque, molto probabile che la quinta generazione determinerà un ulteriore scambio di posti, a seconda di quando e come i vari gruppi di tecnologia dell'informazione sfrutteranno i progressi della nuova tecnologia. La maggior parte dei programmi di sviluppo nazionali sono soggetti a limitazioni sul trasferimento dei risultati delle ricerche alle società dello stesso paese o regione, ma la validità di questa politica è ancora da verificare. Come sempre, i paesi del blocco orientale sono più indietro rispetto alla controparte occidentale, ma si stanno sforzando di tenere il passo con ogni mezzo a loro disposizione.
Va sottolineato che, per la prima volta nella storia, il Giappone è passato in testa: il nome, il concetto e il cammino di sviluppo dei computer della quinta generazione sono tutti frutto dell'iniziativa giapponese. Per quanto attualmente non ci sia una chiara classifica dei progressi compiuti dai gruppi di sviluppo, il profondo impegno con cui sicuramente il Giappone si applicherà in queste ricerche unitamente al suo ruolo, ormai già consolidato, di realizzatore e perfezionatore di tecnologie già stabilite, potrebbero essere decisivi. Gli Stati Uniti, con la loro profonda esperienza nel settore dell'intelligenza artificiale e i loro solidi istituti di tecnologia dell'informazione, si trovano in posizione di vantaggio. È significativo il fatto che l'IBM non si sia pubblicamente impegnata a svolgere attività sulla quinta generazione. Questa è ormai divenuta la prassi tradizionale: l'IBM, infatti, ha sempre lasciato agli altri la parte dell'innovatore, ed è entrata sullo scenario commerciale in un secondo momento. Con effetti sorprendenti, questo è stato il caso dei mainframe negli anni cinquanta e dei microcomputer negli anni ottanta, e potrebbe essere anche il caso dei sistemi intelligenti negli anni novanta. In Gran Bretagna e in Europa, la situazione è molto più in bilico. Dall'inizio degli anni ottanta la Gran Bretagna ha subito una diminuzione graduale delle proprie ricchezze, passando dal ruolo di importante esportatore di prodotti hardware e software a quello di importatore. Per quanto in passato il Regno Unito sia stato un innovatore nel settore informatico, l'andamento degli sviluppi e dello sfruttamento degli stessi non è stato altrettanto prospero e le ricerche sull'intelligenza artificiale in Gran Bretagna hanno avuto un blocco quasi totale negli anni settanta. Nel resto dell'Europa il problema è stabilire se produttori e consumatori nazionali si convinceranno a formare un mercato abbastanza coerente da poter competere, uniti, con gli americani e i giapponesi.

11.5 Intelligenza umana e artificiale

Nel valutare le possibili conseguenze della realizzazione dei sistemi di elaborazione intelligenti, non bisogna lasciarsi prendere troppo da supposizioni fantascientifiche. Anche se i computer della quinta generazione saranno, per certi aspetti, molto più intelligenti dei computer di oggi, non arriveranno mai, in termini generali, neppure in prossimità dei livelli d'intelligenza umana. I computer della quinta generazione non riusciranno a superare il test di Turing, ed è improbabile che riescano a realizzare la massima aspirazione dell'intelligenza artificiale: automatizzare il buon senso umano.

Lo scopo degli elaboratori elettronici della quinta generazione non è quello di rimpiazzare l'intelligenza umana, ma di esserne complementari, anche se a un livello superiore rispetto al passato. L'interazione tra un computer e il suo utente passerà a un più alto livello concettuale — in molti casi un vasto sottoinsieme del linguaggio naturale — ma l'utente sarà consapevole di utilizzare un computer e non di interagire con una persona. Alla parte elettronica dell'interfaccia utente viene trasferito un certo grado di intelligenza, ma l'analisi approfondita, la comprensione, l'ispirazione, la creatività e l'iniziativa sono ancora stretto dominio dell'utente. Comunque, per utilizzare nel modo migliore i sistemi di elaborazione intelligenti, si richiede che l'utente, a sua volta, li sappia usare in modo intelligente.

I computer hanno coperto una certa distanza, passando attraverso le tappe del calcolo, dell'ordinamento, della selezione e della complessa combinazione di decisioni binarie. Attualmente stanno entrando nella regione della competenza specialistica in settori particolari, che consiste nel trarre inferenze partendo dalle basi di conoscenza e fornire consigli ragionati. Ma la distanza coperta è minima se confrontata con quella che rimane da percorrere, ed il cammino da seguire non è ben demarcato. Non è neppure chiaro se i computer, così come li intendiamo al giorno d'oggi o come stanno evolvendo, siano il veicolo adatto per continuare a percorrere il cammino che ci ha condotto qui. La situazione attuale viene ben riassunta nelle parole del giornalista di tecnologia informatica Rex Malik: un computer è un "amplificatore dell'intelligenza attiva". Questa situazione è destinata a rimanere tale per un periodo futuro che va oltre la durata della quinta generazione.

11.6 Conclusione

Il problema centrale è stabilire se si raggiungeranno gli obiettivi della quinta generazione. La questione, per il momento, è destinata a rimanere aperta, ma l'opinione comune è che la direzione in cui si svolgono le ricerche sia quella giusta: "il cammino verso i Sistemi della Quinta Generazione riserverà certamente molti rischi e molti fallimenti, ma non c'è altra strada" (Feigenbaum, 1982). È significativo il fatto che tutti i primi obiettivi del programma giapponese siano stati raggiunti nel tempo stabilito o anche prima, e che non ci sono state defezioni da parte di nessuno dei partecipanti. Il reclutamento di ingegneri della conoscenza e di personale addestrato o con esperienza nel settore dell'intelligenza artificiale sta aumentando notevolmente in ogni campo della tecnologia dell'informazione e si cominciano a vedere sul mercato i primi prodotti hardware e software con le capacità dei sistemi esperti o dell'intelligenza artificiale. L'iniziativa della quinta generazione ha fornito a tutta la comunità della tecnologia dell'informazione gli obiettivi e l'impulso di cui aveva bisogno — "a dir poco, il Giappone ha stabilito gli obiettivi del mondo informatico per il resto di questo decennio e oltre" (Bob Muller, SPL International).

Si possono identificare chiaramente alcuni degli ostacoli che i gruppi di sviluppo della quinta generazione dovranno superare. Il più difficile è costituito, certamente, dalla rappresentazione della conoscenza in generale. Se si riuscirà a stabilire una solida base per le tecniche di rappresentazione della conoscenza, allora sarà molto più facile superare alcuni degli ostacoli ad essa associati, come l'interpretazione del linguaggio naturale e il riconoscimento della voce. Per quanto riguarda la progettazione dei sistemi, l'ostacolo principale è riuscire a realizzare un'architettura generale adatta ai sistemi di elaborazione parallela e, di conseguenza, trovare il linguaggio di programmazione (o l'insieme di linguaggi di programmazione) che permetta di lavorare in modo efficiente sulle architetture parallele e, allo stesso tempo, di soddisfare i requisiti logici dei sistemi intelligenti basati sulla conoscenza. L'ostacolo forse minore è la necessità di aumentare di almeno due ordini di grandezza il numero di elementi presenti sui chip e la massima velocità di elaborazione raggiungibile. L'ultimo ostacolo riguarda le applicazioni: sfruttare le capacità dei sistemi di elaborazione intelligenti per riuscire a realizzare, negli anni novanta, sistemi informativi produttivi, vantaggiosi e con costi contenuti. Per quanto la meta finale di un computer della quinta generazione sia unica e chiara-

mente identificata, si possono scegliere diverse direzioni di ricerca per raggiungerla. Il piano d'azione più prudente è il metodo, seguito nel programma Alvey, delle tecnologie di attivazione che portano a sistemi intelligenti ma che, per caso o per progetto, possono fornire anche molti vantaggi intermedi. Le ricerche su nuove architetture di computer VLSI e sull'ingegneria del software sono fonte di notevoli vantaggi per l'elaborazione dati tradizionale e per tutti gli altri settori della tecnologia dell'informazione. L'attività sulle basi di conoscenza si riflette positivamente sui database. Ogni altro miglioramento apportato alle interfacce utente è utile ai sistemi di elaborazione dati di tutti i tipi.

In ultima analisi, il successo o il fallimento dell'iniziativa della quinta generazione dipende da quanti saranno i professionisti della tecnologia dell'informazione con una certa conoscenza e comprensione del progetto e di ciò che esso comporta, e in grado di utilizzare i frutti di questi sviluppi non appena siano disponibili. Proprio a questi gruppi è rivolto questo libro, per far sì che la tecnologia dell'informazione mantenga la rotta che la porterà ad essere il maggiore comparto industriale del mondo entro la fine del secolo.

Glossario

algoritmo Descrizione dei passi necessari per la soluzione di un dato compito.

allofono Componente elementare del suono di un linguaggio

altissima integrazione (VLSI) Inclusione di un numero elevatissimo (decine di migliaia o più) di elementi discreti su di un singolo circuito integrato.

ambiente integrato di supporto alla programmazione (*Integrated programming support environment*, Ipse) Insieme compatibile di strumenti basati su una metodologia di supporto delle attività sia tecniche che gestionali in tutte le fasi di operazione e sviluppo del software.

analisi sintattica Determinazione della struttura sintattica di un programma o di un testo in un linguaggio naturale.

apprendimento assistito dall'elaboratore (*Computer Aided Learning*, CAL) Impiego dei computer a scopi didattici

approccio dell'elaboratore di stato Tecnica di progettazione del software per mezzo della rappresentazione del compito da eseguire come macchina a stati finiti..

architettura a flusso Disposizione in rete di unità di elaborazione parallele all'interno di un computer in modo che i dati fluiscano da un elemento all'altro durante l'esecuzione di un programma

architettura a riduzione dei grafi Disposizione degli elementi di elaborazione paralleli in un computer in modo che essi eseguano direttamente la struttura del grafo di un programma.

base di conoscenza Patrimonio di conoscenze memorizzate in un sistema di elaborazione della quinta generazione, da cui vengono tratte le inferenze.

calcolo dei predicati Tecnica formale di applicazione delle regole di inferenza ai predicati.

casellario Struttura per la memorizzazione di un nodo in una rete di rappresentazione della conoscenza.

clausola di Horn Tipo di proposizione della logica dei predicati con una o più condizioni concatenate dall'operatore AND, e un'unica conclusione.

computer di prima generazione Computer basato su valvole.

computer di quarta generazione Computer basato su microprocessori e chip LSI

computer di quinta generazione Computer che utilizza inferenze per trarre conclusioni ragionate da una base di conoscenza e che interagisce con il suo utente per mezzo di interfacce intelligenti.

computer di seconda generazione Computer basato su transistor discreti.

computer di terza generazione Computer basato su circuiti integrati (ma non microprocessori).

computer Macchina che, sotto il controllo di un programma memorizzato, automaticamente acquisisce, elabora e fornisce dati e può anche memorizzare, rintrac-

ciare, trasmettere e ricevere dati.

copione o **script** Descrizione del contesto in cui si svolge la comunicazione in linguaggio naturale tra un computer ed il suo utente

database relazionale Base di dati strutturata come insiemi di esempi di relazioni tra singoli dati.

Defense Advanced Research Project Agency (Darpa) Branca del Dipartimento americano della difesa responsabile dei progetti della quinta generazione.

divario semantico Divario esistente tra le capacità di basso livello dell'hardware di un elaboratore ed i requisiti applicativi d'alto livello cui deve rispondere il suo software.

elaborazione dell'immagine Elaborazione, da parte del computer, di immagini visive simili a quelle prodotte da una cinepresa.

elaborazione parallela Esecuzione contemporanea di più di una sequenza di passi procedurali all'interno di uno stesso sistema di elaborazione dati.

Entscheidungsproblem (problema della decisione) Problema consistente nello stabilire se esiste un metodo preciso che, applicato a qualsiasi problema matematico, determini se questo abbia o meno una soluzione.

esplosione combinatoria Situazione determinantesi quando il numero di passi procedurali necessari all'esecuzione di un'operazione, quale una ricerca, supera di molto la capacità del computer di effettuare l'elaborazione in tempi accettabili.

Esprit Programma della CEE per lo sviluppo dei computer della quinta generazione.

euristico Secondo metodi dettati dal buon senso o regole empiriche.

evoluzione del software Punto di vista olistico del ciclo di stadi nella vita di un

oggetto software: progettazione, sviluppo, commissione, assistenza e revisione.

fabbrica di sistemi informativi Impianto per la progettazione e lo sviluppo di computer con hardware e software integrato.

finestra Porzione dello schermo video di un computer che permette di visualizzare l'output di un programma mentre sul resto dello schermo viene visualizzato un altro programma in esecuzione.

flusso di controllo parallelo Passaggio del controllo da un punto centrale a due o più processi sequenziali paralleli in un sistema di computer paralleli.

fonderia di silicio Insieme integrato di dispositivi per la progettazione assistita dall'elaboratore e la fabbricazione dei chip ad altissima integrazione.

frattale Tipo di funzione matematica consistente in una sequenza di linee discontinue o segmenti di aree, ricorrenti all'interno di ogni linea o area.

gigabyte Un miliardo di byte.

guscio di sistema esperto Struttura vuota di un sistema esperto in cui viene costruito un certo campo di competenza umana.

hardware Componenti fisiche di un sistema di computer.

icona Simbolo visivo che appare sullo schermo video di un computer quale parte della sua interfaccia utente.

indipendenza dei dati Separazione dei modelli di dati logici di un database dalla struttura fisica dei dati memorizzati.

induzione matematica Tecnica di dimostrazione di alcuni tipi di teoremi matematici secondo cui, se il teorema è vero in un caso particolare, e allora esso è vero anche nel caso "seguente", è vero in tutti i casi.

inferenze logiche al secondo (*logical inferences per second*, lips) Misura della velocità di prestazione di un sistema di elaborazione dati della quinta generazione.

ingegnere della conoscenza Chi progetta sistemi esperti o sistemi intelligenti basati sulla conoscenza.

ingegneria del software Professione volta alla progettazione, sviluppo e assistenza al software di un computer in modo organizzato, soggetto a vincoli di costo, tempo e prestazione.

Institute for New Generation Computer Technology (Icot) Il centro del programma giapponese di sviluppo dei computer della quinta generazione.

integrazione a wafer Fabbricazione di un singolo chip di 5 cm quadri dotato di numerosi elementi discreti, su un substrato.

intelligenza artificiale Capacità del computer di comportarsi in modo tale che, se si trattasse del comportamento di una persona, verrebbe considerato intelligente.

interfaccia utente Punto di contatto tra un sistema di elaborazione dati e la persona che lo utilizza.

lambda calcolo Metodologia basata su una notazione formale per l'espressione di enunciati logici o matematici.

linguaggi di programmazione non procedurale Classe di linguaggi di programmazione comprendente i linguaggi dichiarativi ed applicativi che specifica quali operazioni di elaborazione sono necessarie senza però specificare come eseguire i passi procedurali.

linguaggio di programmazione applicativo Linguaggio di programmazione in cui i programmi vengono strutturati come funzioni operanti su tipi di dati astratti e privo di attribuzioni distruttive.

linguaggio di programmazione dichiarativo Linguaggio di programmazione

che consente di descrivere i compiti in base alle strutture di dati da utilizzare e specifica quali sono le operazioni di elaborazione da effettuare sui dati anziché il modo in cui svolgere l'elaborazione, come accade nei linguaggi procedurali.

linguaggio di programmazione procedurale Linguaggio di programmazione che richiede che i passi dettagliati dell'elaborazione dei dati vengano specificati per ogni compito.

macchina di Turing Elaboratore astratto progettato da Alan Turing che può, in linea di principio, risolvere qualsiasi problema espresso per mezzo di un algoritmo.

matrice di elaborazione Vettore o matrice di elementi d'elaborazione identici operanti tutti in modo sincrono.

matrice sistolica Assemblaggio di processori speciali su unico chip progettati per eseguire tipi particolari di algoritmi di elaborazione.

megabyte Un milione di byte.

memoria ad accesso diretto (*Direct Memory Access*, DMA) Accesso diretto alla memoria centrale dell'elaboratore dai dispositivi periferici, bypassando i registri di elaborazione.

Microelectronic and Computer Technology Corporation (MCC) Società sponsorizzata da un consorzio di società di tecnologia dell'informazione americane che svolge ricerca e sviluppo sulla quinta generazione per conto dei suoi sponsor.

micrometro Un milionesimo di metro.

minimax Strategia di gioco consistente nel minimizzare il massimo vantaggio possibile che l'avversario può trarre, durante il gioco, in seguito ad una particolare mossa.

Ministry of International Trade and Industry (MITI) ramo dell'amministrazione civile giapponese responsabile dello sviluppo industriale.

modulo Sistema con limiti ben definiti e interfacce esterne standardizzate

motore inferenziale Elemento di un sistema di computer della quinta generazione che trae conclusioni ragionate partendo dalla conoscenza.

mouse Oggetto a presa manuale che, spostato su di una superficie piatta, fa muovere un puntatore in direzione corrispondente sullo schermo di un computer.

nanosecondo Un miliardesimo di secondo.

parallelismo regolare Elaborazione parallela svolta da processori identici operanti in modo sincrono

perfezionamento graduale (*stepwise refinement*) Tecnica di progettazione del software che comincia dalla descrizione di alto livello del compito da eseguire e continua espandendo gli stadi in modo sempre più dettagliato.

pipeline Insieme di elementi di elaborazione dedicati connessi l'uno all'altro in sequenza, ognuno dei quali esegue un passo di una particolare operazione.

predicato Relazione logica.

processi sequenziali comunicanti Insieme di compiti, ognuno svolto da un processore sequenziale di un computer, che, in determinati stadi, può trasferire le informazioni da un processore all'altro.

processore sequenziale delle inferenze per uso personale (PSI) Stazione di lavoro basata sul Prolog usata dai ricercatori del centro Icot.

progettazione assistita dall'elaboratore (*Computer Aided Design*, CAD) Uso di strumenti computerizzati per la progettazione e il disegno.

programma Alvey Programma inglese per lo sviluppo dei computer della quinta generazione.

programma Insieme di istruzioni per controllare il funzionamento del computer.

ragionamento basato sulle evidenze Analisi di prove, quali quelle presentate in un processo in tribunale, effettuata per raggiungere l'unanimità di opinione e per trarre conclusioni generali.

rappresentazione della conoscenza dichiarativista Rappresentazione della conoscenza indipendentemente dalle procedure di controllo utilizzate nell'elaborazione della conoscenza.

rappresentazione della conoscenza proceduralista Tecnica di rappresentazione della conoscenza in cui le strutture di controllo per elaborare la conoscenza sono parte integrante della conoscenza stessa.

regola di inferenza Regola generale utilizzata nel calcolo dei predicati.

rete semantica (*semantic network*) Tecnica di rappresentazione della conoscenza per mezzo di nodi, indicanti gli oggetti, e archi, indicanti le relazioni esistenti tra essi.

riconoscimento della voce Capacità di un sistema di elaboratori di riconoscere ed interpretare un input costituito dal parlato.

scomposizione funzionale Specifica della struttura di un programma che comincia dalla funzione di livello più alto per poi scomporla in insiemi di funzioni di livello inferiore che assieme realizzano la definizione delle funzioni di livello superiore.

semantica Studio del significato.

sintassi Struttura di un linguaggio naturale o di un linguaggio di programmazione conforme ad un insieme di regole formali.

sintesi del parlato Produzione di suoni vocali da parte di un computer partendo da un testo memorizzato al suo interno.

sistema di produzione Insieme di enunciati che descrivono le proprietà logiche di una base di conoscenza.

sistema di supporto decisionale Sistema di computer utilizzato dai manager, operante sulla base di conoscenza di una data organizzazione che, per coadiuvare le decisioni organizzative, fornisce dei consigli ragionati.

sistema esperto Sistema di computer che automatizza un certo grado di competenza umana in un determinato settore.

sistema Insieme di elementi correlati compresi entro limiti ben precisi che assieme raggiungono, o cercano di raggiungere, scopi ed obiettivi specifici.

sistema intelligente basato sulla conoscenza (*Intelligent Knowledge Based System*, IKBS) Sistema di elaborazione dati che, per eseguire un compito, applica procedimenti di inferenza alla conoscenza.

software Programmi che controllano il funzionamento dell'elaboratore.

spazio degli stati (o **spazio di ricerca**) Insieme degli stati da generare e verificare o esaminare durante una ricerca.

tecnologia di attivazione Una delle quattro tecnologie (VLSI, sistemi intelligenti basati sulla conoscenza, ingegneria del software e interfacce utente intelligenti) che apre la strada ai sistemi di elaborazione dati della quinta generazione.

tipo di dati astratto Classe di dati di elaborazione definita in base alle proprie caratteristiche logiche e indipendentemente dalla propria rappresentazione fisica in un sistema di elaborazione dati.

transputer Processore *single-chip*, con memoria incorporata, dotato di elevatissima capacità di trattamento dei dati, parte di un'architettura di processori paralleli.

Bibliografia

Aiso, Hideo (1982), "Fifth generation computer architecture" in Moto-Oka (1982), pp. 121-127.

Allen, J. (1983), "VLSI applications: Speech processing", in Scarrott (1983), pp.41-48.

Allen, Johnatan (1982), "Algorithms, Architecture and Technology", in Moto-Oka (1982), pp.277-281.

Alty, J. L. e Coombs, M. J. (1984), *Expert Systems: Concepts and Examples*, National Computing Centre.

Alvey, John (1982), *A Programme for Advanced Information Technology*, HMSO.

Appel, K., Haken, W. (1976), "Every planar map is four colourable", *Bulletin of the American Mathematical Society*, 82, 5, 711-712.

Ashby, W.Ross (1952), *Design for a Brain*, Chapman & Hall (trad. it. *Progetto per un cervello*, Bompiani, Milano, 1970).

Ashurst, F.Gareth (1983), *Pioneers of Computing*, Frederick Muller.

Babbage, Charles (1837), "On the mathematical powers of the calculating engine", in Randell (1973), pp.19-54.

Babich, Wayne a> (1985), *Software Configuration Management: Co- ordination and Control for Productivity*, Addison-Wesley.

Bailey, Roger (1985), "A short tutorial on the programming language Hope", Departmente of Computing, Imperial College.

Bailey, Roger (1986), *Functional Programming with Hope*, Ellis Horwood.

Barnes, John (1982), *Programming in Ada*, Addison-Wesley.

Bauer, F. L. (a cura di) (1973), *Software Engineering: An advanced Course*, Springer Verlag.

Beishon, John e Peters, Geoffrey (1972), *Systems Behaviour*, Harper & Row, per Open University Press.

Boden, Margaret A. (1977), *Artificial Intelligence and Natural Man*, Harvester Press (seconda edizione riveduta e ampliata 1986, MIT Press).

Boole, George (1848), *The Mathematical Analysis of Logic*, Dover Publications, New York.

Boole, George (1854), *An Investigation on the Laws of Thought, on which are founded the Mathematical Theories of Logic and Probabilities*, Dover Publications, New York.

Bowden, B.V. (a cura di) (1953), *Faster than Thought*, Pitman.

Boyer, R.S., e Moore, J. Strother (a cura di), (1981), *The Correctness Problem in Computer Science*, Academic Press.

Brady, J.M. (1977), *The Theory of Computing Science*, Chapman & Hall.

Bramer, Max (1984), *Fifth Generation: An Annotated Bibliography*, Addison-Wesley.

Brooks, Frederick p. (1975), *The Mythical Man-Month: Essays on Software Engineering*, Addison-Wesley.

Bruckert, E., Minow, M., e Tetschner, W. (1983), "Three-tiered software and VLSI aid developmental system to read text aloud", *Electronics* **55**, 8, 133-138.

Burks, A.W., Goldstine, H.H., e Von Neumann, John (1946), "Preliminary discussion of the logical design of a computing instrument", in Swartlander (1976), pp.221-259, e Randell (1973), pp.399-414.

Campbell, John (a cura di) (1984), *Implementation of Prolog*, Ellis Horwood.

Chomsky, Noam (1965), *Aspects of the Theory of Syntax*, MIT Press.

Chomsky, Noam, (1968), *Language and Mind*, Harcourt, Brace & World.

Church, Alonzo (1936), "A note on the Entscheidungsproblem/, *Journal of Symbolic Logic*, **1**; ristampato a Davis (1965).

Cripps, Martin, Field, A.J., and Reeve, M.J. (1985), "The design and implementation of Alice: a parallel graph reduction machine", Department of Computing, Imperial College; ristampato in *Byte Magazine*, June1985.

Darlington, John, e Reeve, M.J. (1981), "Alice: a multiprocessor reduction machine for the parallel evaluation of applicative languages", *ACM Conference on Functional Programming and Computer Architecture*, October 1981, pp.65-75.

Davis, Martin (a cura di) (1965) *The Undecidable*, Raven Press.

De Bono, Edward (1969), *The Mechanism of Mind*, Penguin Books.

De Bono, Edward (1985), "Starteck: thinking about self-organising systems", in *Computing: The Magazine*, 25th April 1985.

De Saram, Hugh (1985), *Programming in micro-Prolog*, Ellis Horwood.

Duff, M. J. B. (1982), "Parallel architecture and vision ", in SPL (1982).

Ennals, Richard (1983), *Beginning micro-Prolog*, Ellis Horwood.

Erman, L.D., Hayes-Roth, F., Lessere, V.R., e Raj Reddy, D. (1980), "The Hearsay II speech understanding system: integrating knowledge to resolve uncertainty", *Computing Survey* 12, 2, 213-253, June 1980.

Feigenbaum, Edward (1982), "Innovation and symbol manipulation in fifth generation computers" in Moto-Oka (1982), pp.223-226.

Foster, M.J., e Kung, H.T. (1980), "The design of special-purpose VLSI chips", *IEEE Computer Magazine* 13, 1, 26-40.

Fuche, K. (1983), "The direction the FGCS project will take", *New Generation Computing* 1,1,3-9.

Furukawa, K., Nakajima, R., Yonezawa, A., Goto, S., e Aoyama, A. (1982), "Problem solving and inference mechanisms" in Moto-Oka (1982), pp.131-138.

Gannon, T.F. (1983), "Background paper on the Microelectronics and Computer Technology Corporation (MCC)", in SPL International (1983).

Goldstine, Herman H. (1972), *The Computer from Pascal to Von Neumann*, Princeton University Press (trad. it. *Il computer da Pascal a von Neumann*, Etas, Milano, 1981).

Gurd, John (1982), "Developments in dataflow architecture", in SPL International, 1982.

Harris, L. (1982), "The four obstacles to end user computer access", in SPL International (1982).

Hawley, r. (a cura di) (1986), *Artificial Intelligence Programming Environment*, Ellis Horwood.

Hendrix, G., e Sacerdoti, E. (1981), "Natural-language processing: the field in perspective", *Byte* **6**, **9**, 304- 352.

Hoare, C. A. R. (1978), "Communicating sequential processes", *Communications of the ACM* **21**, 8, 666-677.

Hodges, Andrew, (1983), *Alan Turing: The Enigma of Intelligence*, Unwin.

Hofstadter, D., Dennett, D. C. (a cura di) (1981), *The Mind' s I*, Harvester Press (trad. it. *L'Io della mente*, Adelphi, Milano, 1985).

Hyman, Anthony (1982), *Charles Babbage: Pioneer of the Computer*, Oxford University Press.

Inmos (1984), *Occam Programming Manual*, Prentice-Hall.

JIPDEC (1981), "Preliminary report on study and research on Fifth-generation computers", Japan Information Processing Development Centre, in Moto-Oka (1982), pp.3-89.

JIPDEC (1981), "Preliminary report on study and research on fifth generation computers", Japan Information Processing Development Centre, in Moto-Oka (1982), pp.2-89.

Kaplan, S. J., e Ferris, D. (1982), "Natural language in the DP world", *Datamation* **28**, 9, 114-120.

Kikuchi, Ahira (a cura di)(1983), "Discussion at the start of the fifth generation computer systems project", *Icot Journal* No. 1.

Kowalski, Robert (1982), "Logic as a computer language in education", in Steels and Campbell (1985), pp.71-92.

Kowalski, Robert (1984), "Logic as a database language", Imperial College Research Report DoC 82/25.

Langley, P. (1985), "Strategy acquisition governed by experimentation", in Steels e Campbell (1985), pp.52-68.

Lehman, Meir M. (1980), "Programs, life cycles and laws of software evolution", *Proceedings of the IEEE*, **86**, 9.

Lehman, Meir M. (1982), "Program evolution ", Imperial College Research Report DoC 82/1, December 1982.

Lehman, Meir M. (1984), "Program evolution, programming processes, programming support", Imperial College Research Report DoC 84/1, February 1984.

Lighthill, James (1972), *Artificial Intelligence: Report to SERC*, HMSO.

Lowrance, J.D., e Garvey, T.D. (1982), "Evidential reasoning: a developing concept", *Proceedings of the International Conference on Cybernetics and Society, Seattle, 1982*, pp.6-9, IEEE.

McCarthy, John (1965), *Lisp 1.5 Programmer's Manual*, MIT Press.

McCormick, B. H., Kent, E., e Dyer, C. R. (1982), "A cognitive architecture for computer vision", in Moto-Oka (1982), pp.245- 264.

Mead. C. A., e Conway, L. A. (1980), *Introduction to VLSI Systems*, Addison-Wesley.

Menabrea, L.F. (1843), "Sketdh of the analytical engine", tradotto con note aggiunte da Ada Byron. *Scientific Memoirs*, **3**, pp. 666-731. Note ristampate in Bowden (1953).

Michie, Donald (1982), "Aspects of the fifth generation: the Japanese Knowledge bomb", in SPL International (1982).

Minsky, M. L. (a cura di) (1968), *Semantic Information Processing*, MIT Press.

Moto-Oka, T. (a cura di) (1982), *Fifth Generation Computer Systems: Proceedings of the International Conference on Fifth generation Computer Systems, Tokyo, Japan, October 19-22 1981*, North Holland.

Narayanan, A., e Sharkey, N.E. (1985), *An Introduction to Lisp*, Ellis Horwood.

Popper, Karl (1963), *Conjectures and Refutations: The Growth of Scientific Knowledge*, Routledge & Kegan Paul (trad. it. *Congetture e confutazioni*, Il Mulino, Bologna, 1972).

Queinnec, Christian (1984), *Lisp* (English language edition), Macmillan.

Randell, Brian (a cura di) (1973) *The Origin of Digital Computers: Selected Papers*, Springer-Verlag.

Robinson, J.A. (1965), "A machine oriented logic based on the resolution principle", *Journal of the ACM* **12**, 23-41.

Sakamura, K., Sekino, A., Kodaka, T., Uehara, T., e Aiso, H. (1982), "VLSI and system architecture: the new development of System 5G", in Moto-Oka (1982), pp.189-208.

Sammet, Jean E. (1969), *Programming languages: History and Fundamentals*, Prentice-Hall, pp.405-415 e pp.589-603.

Scarrott, G. C. (a cura di) (1983), *The Fifth Generation Computer Project: State of the Art Report*, Pergamon Infotech.

Seitz, Charles (1980), in *Lambda - the Magazine of VLSI Design*, First quarter.

Sell, P.S. (1982), "New computer applications: user and social acceptability of the fifth generation proposals", in SPL International (1982).

Shannon, Claude (1938), "A symbolic analysis of relay and switching circuits" AIEE Transactions 57, pp. 713-723; ristampato a Swartlander (1976).

Sharp, John A. (1985), *Data flow Computing*, Ellis Horwood.

Smith, Kevin (1983), "New computer breed uses transputers for parallel processing", *Electronics* **56**,4, 67-68.

Sommerville, Ian (1982), *Software Engineering*, Addison- Wesley.

SPL Internationa (1982), *The Fifth Generation - Dawn of the Second Computer Age*.

SPL International (1983), *Fifth Generation World Conference*, 1983.

Steels, L., e Campbell, J. A.(a cura di) (1985), *Progress in Artificial Intelligence*, Ellis Horwood.

Swartlander, Earl E. (a cura di) (1976), Computer Design development: Principal Papers, Hayden.

Tanaka, H., et al.(1982), "The preliminary research on the data flow machine and the data base machine as the basic architecture of Fifth generation computer systems", in Moto-Oka (1982).

Tate Gallery (1983), *Harold Cohen*, Tate Gallery Publications.

Taylor, J.M.(1983), "Intelligent knowledge based systems: a programme for action in the UK", Science and Engineering Research Council/ Department of industry IKBS Architecture Study.

Taylor, John (1983), "Knowledge-based systems in defence: applications and

implications", in SPL International (1983).

Treleaven, Philip (1982), "Fifth generation computer architecture analysis", in Moto-Oka (1982), pp 265-275.

Treleaven, Philip (1983), "VLSI processor architectures", in SPL International (1983).

Turing, Alan (1936), "On computable numbers, with an application to the Entscheidungsproblem", Proceedings of the London Mathematical Society 2, 42, pp.230-265.

Turing, Alan (1950), "Computing machinery and intelligence", in *Mind*, October 1950 (trad. it. "Macchine calcolatrici e intelligenza", in V. Somenzi e R. Cordeschi, *La filosofia degli automi*, Boringhieri, Torino, 1986).

Turner, Raymond (1984), *Logic for Artificial Intelligence*, Ellis Horwood.

Von Neumann, John (1945), "Draft report on the Edvac/, in Randell (1973), pp.383-392.

Von Neumann, John (1951), "The general and logical theory of automata", in *Von Neumann, Collected Works*, Vol. V, Pergamon Press, 1963 (trad. it. parziale in V. Somenzi e R. Cordeschi, *La filosofia degli automi*, Boringhieri, Torino, 1986).

Winograd, Terry (1972), *Understanding Natural Language*, Academic Press.

Yasukawa, H. (1983), "LFG in Prolog: toward a formal system for representing grammatical relations", Icot Technicale Report No. TR-019, 1983.

Young, S.J.(1984), *An Introduction to Ada*, seconda edizione riveduta, Ellis Horwood.

Indice analitico

Finito di stampare nel mese di marzo 1988
presso Lito Velox - Trento
Printed in Italy

franco muzzio & c. editore

A cavallo fra gli anni settanta e gli ottanta, smentendo lo stereotipo che voleva la sua industria abile solo nello sfruttare e perfezionare (o magari copiare) tecnologie nate altrove, il Giappone ha lanciato un piano per la realizzazione di una nuova generazione di calcolatori, più potenti e soprattutto più intelligenti. Questo piano, promosso e coordinato a livello nazionale attraverso il MITI, il ministero giapponese per il commercio, ha fatto discutere molto, sul piano politico ed economico, poiché apre una corsa alla leadership in un settore trainante come quello della tecnologia dell'informazione, e ha rapidamente provocato la formulazione di progetti concorrenti negli Usa e in Europa.
Al di là delle implicazioni politiche ed economiche, i progetti per la quinta generazione condividono una sostanza scientifica e tecnologica di non poco conto: è questo l'argomento del volume di Peter Bishop, che fa opera di accorta divulgazione presentando, per tutti coloro che sono coinvolti nel mondo dell'informatica (docenti, professionisti, tecnici, studenti e manager), le idee di fondo e gli strumenti di cui dispone quel manipolo (neanche tanto piccolo) di ricercatori che è attivamente impegnato nel dar vita alla generazione di calcolatori degli anni ottanta (e forse oltre).

L. 30.000

ISBN 88-7021-417-6

peter bishop

I CALCOLATORI DELLA QUINTA GENERAZIONE